5
Introducing Bitcoin

Bitcoin is the first application of blockchain technology. In this chapter, you will be introduced to Bitcoin technology in detail.

Bitcoin has started a revolution with the introduction of the very first fully decentralized digital currency, and the one that has proven to be extremely secure and stable from a network and protocol point of view. As a currency bitcoin is quite unstable and highly volatile, albeit valuable. We will explain this later in the chapter. This has also sparked a great interest in academic and industrial research and introduced many new research areas.

Since its introduction in 2008 by Satoshi Nakamoto, Bitcoin has gained massive popularity, and it is currently the most successful digital currency in the world with billions of dollars invested in it. The current market cap, at the time of writing, for this currency is $149, 984, 293, 122. Its popularity is also evident from the high number of users and investors, increasing bitcoin price, everyday news related to Bitcoin, and the number of start-ups and companies that are offering bitcoin-based online exchanges, and it's now also traded as *Bitcoin Futures* on **Chicago Mercantile Exchange** (**CME**).

Interested readers can read more about *Bitcoin Futures* at `http://www.cmegroup.com/trading/bitcoin-futures.html`.

The name of the Bitcoin inventor *Satoshi Nakamoto* is believed to be a pseudonym, as the true identity of Bitcoin inventor is unknown. It is built on decades of research in the field of cryptography, digital cash, and distributed computing. In the following section, a brief history is presented in order to provide the background required to understand the foundations behind the invention of Bitcoin.

Digital currencies have always been an active area of research for many decades. Early proposals to create digital cash go as far back as the early 1980s. In 1982, David Chaum, a computer scientist, and cryptographer proposed a scheme that used blind signatures to build untraceable digital currency. This research was published in a research paper, *Blind Signatures for Untraceable Payments*.

Interested readers can read the original research paper which David Chaum describes his invention of the cryptographic primitive of blind signatures at `http://www.hit.bme.hu/~buttyan/courses/BMEVIHIM219/2009/Chaum.BlindSigForPayment.1982.PDF`.

In this scheme, a bank would issue digital money by signing a blind and random serial number presented to it by the user. The user could then use the digital token signed by the bank as currency. The limitation of this scheme was that the bank had to keep track of all used serial numbers. This was a central system by design and required to be trusted by the users.

Later on, in 1988, David Chaum and others proposed a refined version named e-cash that not only used a blinded signature, but also some private identification data to craft a message that was then sent to the bank.

Original research paper for this is available at `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.26.5759`.

This scheme allowed the detection of double spending but did not prevent it. If the same token was used at two different locations, then the identity of the double spender would be revealed. e-cash could only represent a fixed amount of money.

Adam Back, a cryptographer and now CEO of Blockstream, who is involved in blockchain development, introduced *hashcash* in 1997. It was originally proposed to thwart email spam. The idea behind hashcash was to solve a computational puzzle that was easy to verify but comparatively difficult to compute. The idea was that for a single user and a single email, the extra computational effort was negligible, but someone sending a large number of spam emails would be discouraged as the time and resources required to run the spam campaign would increase substantially.

In 1998, B-money was proposed by Wei Dai, a computer engineer who used to work for Microsoft, which introduced the idea of using **Proof of Work** (**PoW**) to create money. The term *Proof of Work* emerged and got popular later with Bitcoin, but in Wei Dai's B-money a scheme of creating money was introduced by providing a solution to a previously unsolved computational problem. It was referred in the paper as *solution to a previously unsolved computational problem*. This concept is similar to PoW, where money is created by broadcasting a solution to a previously unsolved computational problem.

The original paper is available at `http://www.weidai.com/bmoney.txt`.

A major weakness in the system was that an adversary with higher computational power could generate unsolicited money without allowing the network to adjust to an appropriate difficulty level. The system lacked details on the consensus mechanism between nodes and some security issues such as Sybil attacks were also not addressed. At the same time, Nick Szabo, a computer scientist introduced the concept of BitGold, which was also based on the PoW mechanism but had the same problems as B-money with the exception that the network difficulty level was adjustable. Tomas Sander and Amnon Ta-Shma from the **International Computer Science Institute** (**ICSI**), Berkley introduced an e-cash scheme under a research paper named *Auditable, Anonymous Electronic Cash* in 1999. This scheme, for the first time, used Merkle trees to represent coins and **Zero-Knowledge Proofs** (**ZKPs**) to prove the possession of coins.

The original research paper called *Auditable, Anonymous Electronic Cash* is available at: `http://www.cs.tau.ac.il/~amnon/Papers/ST.crypto99.pdf`.

In this scheme, a central bank was required that kept a record of all used serial numbers. This scheme allowed users to be fully anonymous. This was a theoretical design which was not practical to implement due to inefficient proof mechanisms.

Reusable Proof of Work (**RPoW**) was introduced in 2004 by Hal Finney, a computer scientist, developer and first person to receive Bitcoin from Satoshi Nakamoto. It used the hashcash scheme by Adam Back as a proof of computational resources spent to create the money. This was also a central system that kept a central database to keep track of all used PoW tokens. This was an online system that used remote attestation made possible by a trusted computing platform (TPM hardware).

All the previously mentioned schemes are intelligently designed but were weak from one aspect or another. Specifically, all these schemes rely on a central server that is required to be trusted by the users.

Bitcoin

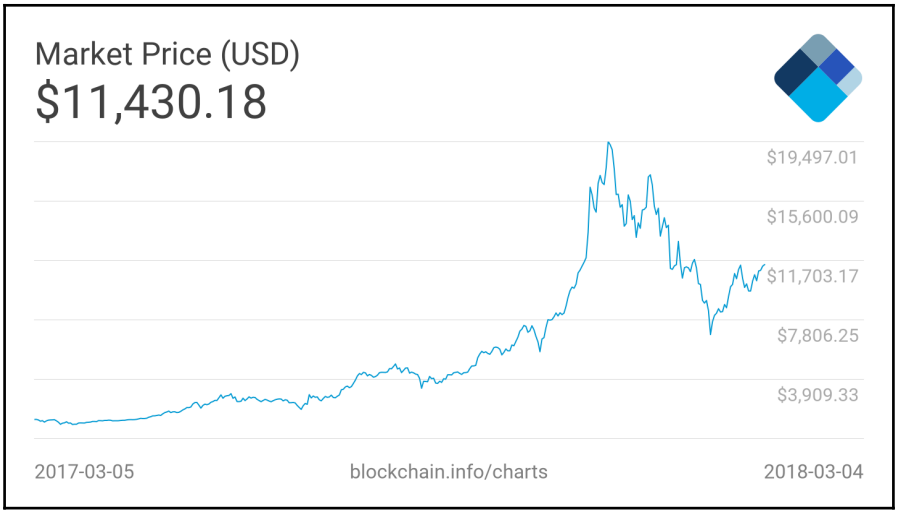
In 2008, Bitcoin was introduced through a paper called, *Bitcoin: A Peer-to-Peer Electronic Cash System*.

This paper is available at `https://bitcoin.org/bitcoin.pdf`.

It was written by Satoshi Nakamoto, which is believed to be a pseudonym, as the true identity of Bitcoin inventor is unknown and subject of much speculation. The first key idea introduced in the paper was of a purely peer-to-peer electronic cash that does need an intermediary bank to transfer payments between peers.

Bitcoin is built on decades of cryptographic research such as the research in Merkle trees, hash functions, public key cryptography, and digital signatures. Moreover, ideas such as BitGold, B-money, hashcash, and cryptographic time stamping provided the foundations for bitcoin invention. All these technologies are cleverly combined in Bitcoin to create the world's first decentralized currency. The key issue that has been addressed in Bitcoin is an elegant solution to the Byzantine Generals' Problem along with a practical solution of the double-spend problem. Recall, that both of these concepts are explained in `Chapter 1`, *Blockchain 101*.

The value of bitcoin has increased significantly since 2011, and then since March 2017 as shown in the following graph:



Bitcoin price since March 2017

The regulation of Bitcoin is a controversial subject and as much as it is a libertarian's dream, law enforcement agencies, governments and banks are proposing various regulations to control it, such as BitLicense issued by New York's state department of financial services. This is a license issued to businesses that perform activities related to virtual currencies. Due to high cost and very strict regulatory requirements pertaining to BitLicense many companies have withdrawn their services from New York.

For people with a libertarian ideology, Bitcoin is a platform which can be used instead of banks for business but they think that because of regulations, Bitcoin may become another institution which is not trusted. The original idea behind Bitcoin was to develop an e-cash system which requires no trusted third party and users can be anonymous. If regulations require **Know Your Customer** (**KYC**) checks and detailed information about business transactions to facilitate regulatory process then it might be too much information to share and as a result Bitcoin may not be attractive anymore to some.

There are now many initiatives being taken to regulate Bitcoin, cryptocurrencies and related activities such as ICOs. **Securities and Exchange Commission** (**SEC**) has recently announced that digital tokens, coins and relevant activities such as **Initial Coin Offerings** (**ICOs**) fall under the category of securities. This means that any digital currency trading platforms will need to be registered with SEC and will have all relevant securities laws and regulations applicable to them. This impacted the Bitcoin price directly and it fell almost 10% on the day this announcement was made.

Interested readers can read more about the regulation of Bitcoin and other relevant activities at `https://www.coindesk.com/category/regulation/`.

The growth of Bitcoin is also due to so-called **network effect**. Also called demand-side economies of scale, it is a concept that basically means more users who use the network, the more valuable it becomes. Over time, an exponential increase has been seen in the Bitcoin network growth. This increase in the number of users is largely financial gain driven. Also, the scarcity of Bitcoin and built-in inflation control mechanism gives it value as there are only 21 million bitcoins that can ever be mined and in addition the miner reward halves every four years. Even though the price of bitcoin fluctuates a lot, it has increased significantly over the last few years. Currently (at the time of writing this), bitcoin price is 9,250 U.S Dollars (USD).

Bitcoin definition

Bitcoin can be defined in various ways; it's a protocol, a digital currency, and a platform. It is a combination of peer-to-peer network, protocols, software that facilitate the creation and usage of the digital currency named bitcoin. Nodes in this peer-to-peer network talk to each other using the Bitcoin protocol.

Note that Bitcoin with a capital B is used to refer to the Bitcoin protocol, whereas bitcoin with a lowercase b is used to refer to bitcoin, the currency.

Decentralization of currency was made possible for the first time with the invention of bitcoin. Moreover, the double spending problem was solved in an elegant and ingenious way in bitcoin. Double spending problem arises when, for example, a user sends coins to two different users at the same time and they are verified independently as valid transactions. The double spending problem is resolved in Bitcoin by using a distributed ledger (blockchain) where every transaction is recorded permanently and by implementing transaction validation and confirmation mechanism. This process will be explained later in the chapter where we introduce the concept of *mining*.

Bitcoin – a bird's-eye view

In this section, we will see how the Bitcoin network looks from a user's point of view. How a transaction is made, how it propagates from a user to the network, how transactions are verified, and finally accumulated in blocks. We will look at what are the various actors and components of the Bitcoin network. Finally, some discussion on how all actors and components interact with each other to form the Bitcoin network will also be provided.

First, let us see that what the main components of a Bitcoin network are. Bitcoin is composed of the elements listed here. We will further expand on these elements as we progress through the chapter.

- Digital keys
- Addresses
- Transactions
- Blockchain
- Miners
- The Bitcoin network
- Wallets (client software)

Now, we will see that how a user will use the Bitcoin network. The following example will help you understand that how the Bitcoin network looks like from an end user's perspective. We will see that what actors and components are involved in a Bitcoin transaction. One of the most common transactions is sending money to someone else, therefore in the following example we will see that how a payment can be sent from one user to another on the Bitcoin network.

Sending a payment to someone

This example will demonstrate that how money can be sent using Bitcoin network from one user to another. There are several steps that are involved in this process. As an example, we are using Blockchain wallet for mobile devices.

The steps are described here:

1. First, either the payment is requested from a user by sending his Bitcoin address to the sender via email or some other means such as SMS, chat applications or in fact any appropriate communication mechanism. The sender can also initiate a transfer to send money to another user. In both cases, the address of beneficiary is required. As an example, the Blockchain wallet is shown here where a payment request is created:

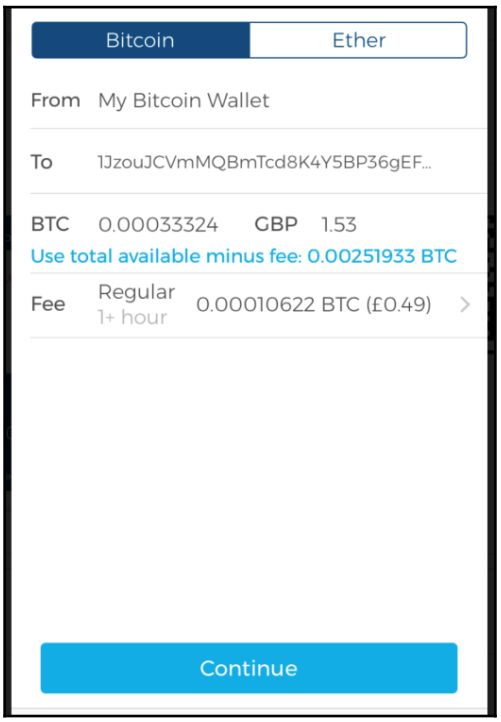
bitcoin payment request (using Blockchain wallet)

2. The sender either enters the receiver's address or scans the QR code that has the Bitcoin address, amount and optional description encoded in it. The wallet application recognizes this QR code and decodes it into something like `Please send <Amount> BTC to the Bitcoin address <receiver's Bitcoin address>`.
3. This will look like as shown here with values: `Please send 0.00033324 BTC to the Bitcoin address 1JzouJCVmMQBmTcd8K4Y5BP36gEFNn1ZJ3`.
4. This is also shown in the screenshot presented here:

Bitcoin payment QR code

The QR code shown in the preceding screenshot is decoded to `bitcoin://1JzouJCVmMQBmTcd8K4Y5BP36gEFNn1ZJ3?amount=0.00033324` which can be opened as a URL in Bitcoin wallet.

5. In the wallet application of the sender, this transaction is constructed by following some rules and broadcasted to the Bitcoin network. This transaction is digitally signed using the private key of the sender before broadcasting it. How the transaction is created, digitally signed, broadcasted, validated and added to the block will become clear in the following sections. From a user's point of view, once the QR code is decoded the transaction will appear similar to what is shown in the following screenshot:

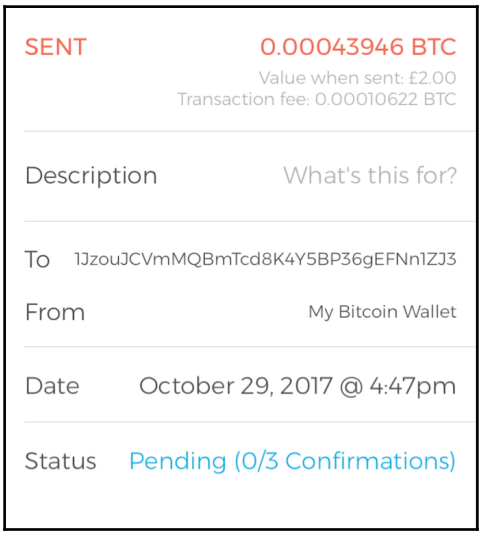


Send BTC using Blockchain wallet

Note that in the preceding screenshot there are a number of fields such as **From**, **To**, **BTC**, and **Fee**. While other fields are self-explanatory, it's worth noting that **Fee** is calculated based on the size of the transaction and a fee rate is a value that depends on the volume of the transaction in the network. This is represented in Satoshis/byte. Fee in Bitcoin network ensures that your transaction will be included by miners in the block.

Recently the Bitcoin fees were so high that even for smaller transactions a high amount of fee was charged. This was due to the fact that miners are free to choose which transactions they pick to verify and add in a block, and they select the ones with higher fees. The high number of users creating thousands of transactions also played a role in causing this situation of high fees because transactions were competing with each other to be picked up first and miners picked up the ones with highest fees. This fee is also usually estimated and calculated by the Bitcoin wallet software automatically before sending the transactions. The higher the transaction fee the more chances are that your transaction will be picked up at priority and included in the block. This task is performed by the miners. Mining and miners is a concept that we will look at a bit later in this chapter in the context of Bitcoin mining.

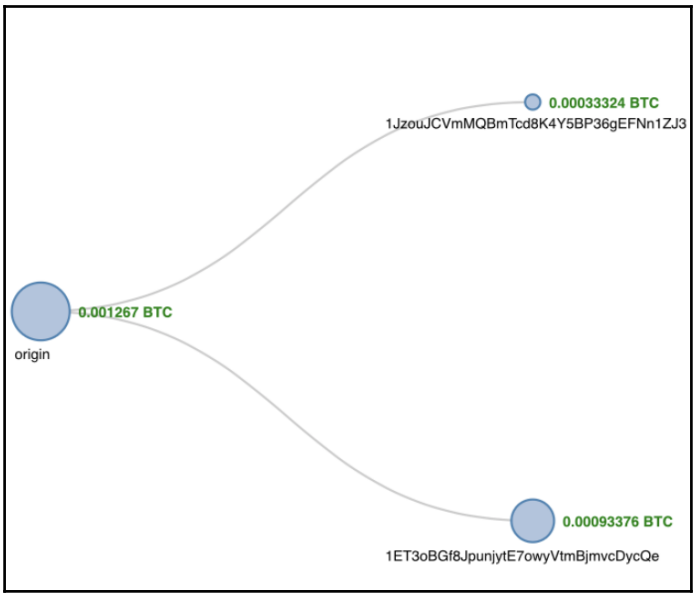
Once the transaction is sent it will appear as shown here in the Blockchain wallet software:



Transaction sent

4. At this stage, the transaction has been constructed, signed and sent out to the Bitcoin network. This transaction will be picked up by miners to be verified and included in the block. Also note that in the preceding screenshot, confirmation is pending for this transaction. These confirmations will start to appear as soon as the transaction is verified, included in the block, and mined. Also, the appropriate fee will be deducted from the original value to be transferred and will be paid to the miner who has included it in the block for mining.

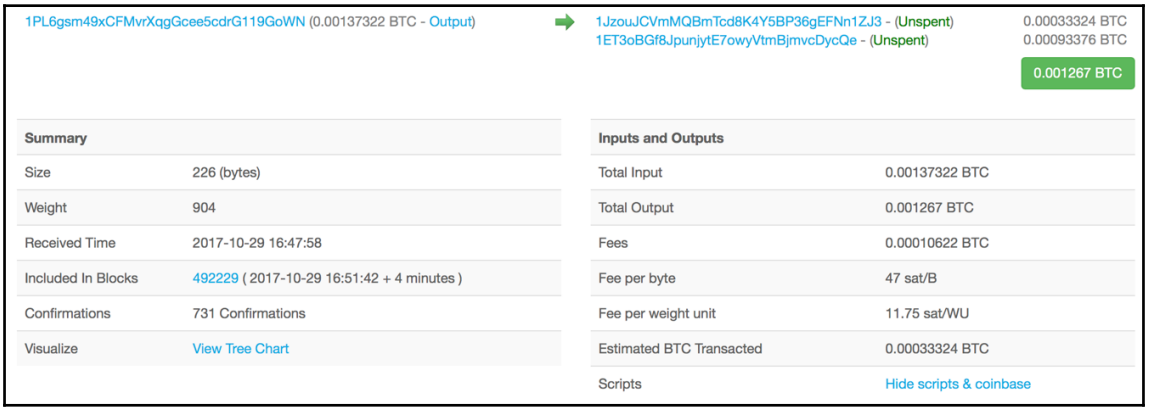
This flow is shown in the following diagram, where a payment of 0.001267 BTC (approximately 11 USD) is originated from the sender's address and been paid to receiver's address (starting with 1Jz). The fee of 0.00010622 (approximately 95 cents) is also deducted from the transaction as mining fee.



Transaction flow visualization (Blockchain.info)

The preceding screenshot visually shows how the transaction flowed on the network from origin (sender) to receivers on the right-hand side.

A summary view of various attributes of the transaction is shown here:



Snapshot of the transaction taken from Blockchain.info

Looking at the preceding screenshot there are a number of fields that contain various values. Important fields are listed here with their purpose and explanation:

- **Size**: This is the size of the transaction in bytes.
- **Weight**: This is the new metric given for block and transaction size since the introduction of **Segregated Witness** (**SegWit**) version of Bitcoin.
- **Received Time**: This is the time when the transaction is received.
- **Included In Blocks**: This shows the block number on the blockchain in which the transaction is included.
- **Confirmations**: This is the number of confirmations by miners for this transaction.
- **Total Input**: This is the number of total inputs in the transaction.
- **Total Output**: This is the number of total outputs in the transaction.
- **Fees**: This is the total fees charged.
- **Fee per byte**: This field represents the total fee divided by the number of bytes in a transaction. For example 10 Satoshis per byte.
- **Fee per weight unit**: For legacy transaction it is calculated using *total number of bytes * 4*. For SegWit transactions it is calculated by combining SegWit marker, flag, and witness field as one weight unit and each byte of other fields as four weight units.

Transaction ID of this transaction on the Bitcoin network is `d28ca5a59b2239864eac1c96d3fd1c23b747f0ded8f5af0161bae8a616b56a1d` and can be further explored using the `https://blockchain.info/tx/d28ca5a59b2239864eac1c96d3fd1c23b747f0ded8f5af0161bae8a616b56a1d` link via services provided by `https://blockchain.info/`. This transaction ID is available in the wallet software after transaction is sent to the network. From there it can be further explored using one of many Bitcoin blockchain explorers available online. We are using `https://blockchain.info/` as an example.

Bitcoin transactions are serialized for transmission over the network and encoded in hexadecimal format. As an example, the preceding transaction, is also shown here. We will see later in the *Transactions* section that how this hexadecimal encoded transaction can be decoded and what fields make up a transaction.

```
01000000017d3876b14a7ac16d8d550abc78345b6571134ff173918a096ef90ff0430e12408
b0000006b483045022100de6fd8120d9f142a82d5da9389e271caa3a757b01757c8e4fa7afb
f92e74257c02202a78d4fbd52ae9f3a0083760d76f84643cf8ab80f5ef971e3f98ccba2c717
58d012102c16942555f5e633645895c9affcb994ea7910097b7734a6c2d25468622f25e12ff
ffffff022c820000000000001976a914c568ffeb46c6a9362e44a5a49deaa6eab05a619a88a
cc06c0100000000001976a9149386c8c880488e80a6ce8f186f788f3585f74aee88ac000000
00
```

In summary, the payment transaction in the Bitcoin network can be divided into the following steps:

1. Transaction starts with a sender signing the transaction with their private key
2. Transaction is serialized so that it can be transmitted over the network
3. Transaction is broadcasted to the network
4. Miners listening for the transactions picks up the transaction
5. Transaction are verified for their validity by the miners
6. Transaction are added to the candidate/proposed block for mining
7. Once mined, the result is broadcasted to all nodes on the Bitcoin network

Mining, transaction and other relevant concepts will become clearer in the following sections in the chapter. Now in the next section various denominations of bitcoin are presented.

The bitcoin currency, being digital has various denominations which are shown in the following table. A sender or receiver can request any amount. The smallest bitcoin denomination is the Satoshi. The bitcoin currency units are described as follows:

DENOMINATION	ABBREVIATION	FAMILIAR NAME	VALUE IN BTC
Satoshi	SAT	Satoshi	0.00000001 BTC
Microbit	µBTC (uBTC)	Microbitcoin or Bit	0.000001 BTC
Millibit	mBTC	Millibitcoin	0.001 BTC
Centibit	cBTC	Centibitcoin	0.01 BTC
Decibit	dBTC	Decibitcoin	0.1 BTC
Bitcoin	BTC	Bitcoin	1 BTC
DecaBit	daBTC	Decabitcoin	10 BTC
Hectobit	hBTC	Hectobitcoin	100 BTC
Kilobit	kBTC	Kilobitcoin	1000 BTC
Megabit	MBTC	Megabitcoin	1000000 BTC

bitcoin denominations

Now you will be introduced to the building blocks of Bitcoin one by one. First, we will look at the keys and addresses which are used to represent the ownership and transfer of value on the Bitcoin network.

Digital keys and addresses

On the Bitcoin network, possession of bitcoins and transfer of value via transactions is reliant upon private keys, public keys, and addresses. In `Chapter 4`, *Public Key Cryptography*, we have already covered these concepts, and here we will see that how private and public keys are used in the Bitcoin network.

Elliptic Curve Cryptography (**ECC**) is used to generate public and private key pairs in the Bitcoin network.

Private keys in Bitcoin

Private keys are required to be kept safe and normally resides only on the owner's side. Private keys are used to digitally sign the transactions proving the ownership of the bitcoins.

Private keys are fundamentally 256-bit numbers randomly chosen in the range specified by the `secp256k1` ECDSA curve recommendation. Any randomly chosen 256-bit number from `0x1` to `0xFFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFE BAAE DCE6 AF48 A03B BFD2 5E8C D036 4140` is a valid private key.

Private keys are usually encoded using **Wallet Import Format** (**WIF**) in order to make them easier to copy and use. It is a way to represent the full size private key in a different format. WIF can be converted into a private key and vice versa. The steps are described here.

The following is an example of a private key:

```
A3ED7EC8A03667180D01FB4251A546C2B9F2FE33507C68B7D9D4E1FA5714195201
```

When it is converted into WIF format it looks like this:

```
L2iN7umV7kbr6LuCmgM27rBnptGbDVc8g4ZBm6EbgTPQXnj1RCZP
```

Interested readers can do some experimentation using the tool available at the following website:
`http://gobittest.appspot.com/PrivateKey`

Also, **mini private key format** is sometimes used to create the private key with a maximum of up to 30 characters in order to allow storage where physical space is limited, for example, etching on physical coins or encoding in damage-resistant QR codes. The QR code becomes more damage resistant because more dots can be used for error correction and less for encoding the private key. The private key encoded using mini private key format is also sometimes called **minikey**. The first character of mini private key is always uppercase letter `S`. A mini private key can be converted into a normal size private key but an existing normal size private key cannot be converted into a mini private key. This format was used in Casascius physical bitcoins.

Interested readers can find more information here
`https://en.bitcoin.it/wiki/Casascius_physical_bitcoins`.

A Casascius physical bitcoin's security hologram paper with minikey and QR code

The Bitcoin core client also allows the encryption of the wallet that contains the private keys.

Public keys in Bitcoin

Public keys exist on the blockchain and all network participants can see it. Public keys are derived from private keys due to their special mathematical relationship with the private keys. Once a transaction signed with the private key is broadcasted on the Bitcoin network, public keys are used by the nodes to verify that the transaction has indeed been signed with the corresponding private key. This process of verification proves the ownership of the bitcoin.

Bitcoin uses ECC based on the `secp256k1` standard. More specifically it makes use of ECDSA to ensure that funds remain secure and can only be spent by the legitimate owner. If you need to refresh the relevant cryptography concepts, you can refer to `Chapter 4`, *Public Key Cryptography* where ECC was explained. A public key is 256-bits in length. Public keys can be represented in an uncompressed or compressed format. Public keys are fundamentally x and y coordinates on an elliptic curve. In an uncompressed format public keys are presented with a prefix of `0x4` in a hexadecimal format. The x and y coordinates are both 32-bit in length. In total, the compressed public key is 33-bytes long as compared to 65-bytes in the uncompressed format. The compressed version of public keys includes only the x part, since the y part can be derived from it.

The reason why the compressed version of public keys works is that if the ECC graph is visualized, it reveals that the *y* coordinate can be either below the *x* axis or above the *x* axis and as the curve is symmetric, only the location in the prime field is required to be stored. If *y* is even then it is above the *x* axis and if it is odd then it is below the *x* axis. This means that instead of storing both *x* and *y* as the public key only *x* can be stored with the information that if *y* is even or odd.

Initially, Bitcoin client used uncompressed keys, but starting from Bitcoin core client 0.6, compressed keys are used as standard. This resulted in almost 50% reduction of space used to store public keys in the blockchain.

Keys are identified by various prefixes, described as follows:

- Uncompressed public keys use `0x04` as the prefix
- Compressed public key starts with `0x03` if the *y* 32-bit part of the public key is odd
- Compressed public key starts with `0x02` if the *y* 32-bit part of the public key is even

Addresses in Bitcoin

A bitcoin address is created by taking the corresponding public key of a private key and hashing it twice, first with the SHA-256 algorithm and then with RIPEMD-160. The resultant 160-bit hash is then prefixed with a version number and finally encoded with a Base58Check encoding scheme. The bitcoin addresses are 26-35 characters long and begin with digit `1` or `3`.

A typical bitcoin address looks like a string shown here:

```
1ANAguGG8bikEv2fYsTBnRUmx7QUcK58wt
```

This is also commonly encoded in a QR code for easy distribution. The QR code of the preceding bitcoin address is shown in the following screenshot:

QR code of a bitcoin address 1ANAguGG8bikEv2fYsTBnRUmx7QUcK58wt

Currently, there are two types of addresses, the commonly used P2PKH and another P2SH type, starting with number `1` and `3`, respectively. In the early days, Bitcoin used direct Pay to Pubkey, which is now superseded by P2PKH. These types will be explained later in the chapter. However, direct Pay to Pubkey is still used in Bitcoin for coinbase addresses. Addresses should not be used more than once; otherwise, privacy and security issues can arise. Avoiding address reuse circumvents anonymity issues to an extent, Bitcoin has some other security issues as well, such as transaction malleability, Sybil attacks, race attacks and selfish mining which require different approaches to resolve.

Transaction malleability has been resolved with so-called *Segregated Witness* soft fork upgrade of the Bitcoin protocol. This concept will be explained later in the chapter.

From bitaddress.org, private key and bitcoin address in a paper wallet

Base58Check encoding

Bitcoin addresses are encoded using the Base58Check encoding. This encoding is used to limit the confusion between various characters, such as 0OIl as they can look the same in different fonts. The encoding basically takes the binary byte arrays and converts them into human-readable strings. This string is composed by utilizing a set of 58 alphanumeric symbols. More explanation and logic can be found in the `base58.h` source file (`https://github.com/bitcoin/bitcoin/blob/master/src/base58.h`) in the bitcoin source code:

```
/**
* Why base-58 instead of standard base-64 encoding?
* - Don't want 0OIl characters that look the same in some fonts and
* could be used to create visually identical looking data.
* - A string with non-alphanumeric characters is not as easily accepted as
input.
* - E-mail usually won't line-break if there's no punctuation to break at.
* - Double-clicking selects the whole string as one word if it's all
alphanumeric.
*/
```

Vanity addresses

As bitcoin addresses are based on base-58 encoding, it is possible to generate addresses that contain human-readable messages. An example is shown as follows:

Vanity public address encoded in QR

Vanity addresses are generated using a purely brute-force method. An example of a paper wallet with vanity address is shown in the following screenshot:

Vanity address generated from https://bitcoinvanitygen.com/

In the preceding screenshot, on the right-hand bottom corner the public vanity address with QR code is displayed. The paper wallets can be stored physically as an alternative to electronic storage of private keys.

Multisignature addresses

As the name implies, these addresses require multiple private keys. In practical terms, it means that in order to release the coins a certain set of signatures is required. This is also known as **M-of-N MultiSig**. Here *M* represents threshold or the minimum number of signatures required from *N* number of keys to release the bitcoins.

Transactions

Transactions are at the core of the bitcoin ecosystem. Transactions can be as simple as just sending some bitcoins to a bitcoin address, or it can be quite complex depending on the requirements. Each transaction is composed of at least one input and output. Inputs can be thought of as coins being spent that have been created in a previous transaction and outputs as coins being created. If a transaction is minting new coins, then there is no input and therefore no signature is needed. If a transaction is to send coins to some other user (a bitcoin address), then it needs to be signed by the sender with their private key and a reference is also required to the previous transaction in order to show the origin of the coins. Coins are, in fact, unspent transaction outputs represented in Satoshis.

Transactions are not encrypted and are publicly visible in the blockchain. Blocks are made up of transactions and these can be viewed using any online blockchain explorer.

The transaction life cycle

The following steps describe the transaction life cycle:

1. A user/sender sends a transaction using wallet software or some other interface.
2. The wallet software signs the transaction using the sender's private key.
3. The transaction is broadcasted to the Bitcoin network using a flooding algorithm.
4. Mining nodes (miners) who are listening for the transactions verify and include this transaction in the next block to be mined. Just before the transaction are placed in the block they are placed in a special memory buffer called **transaction pool**. The purpose of the transaction pool is explained in the next section.
5. Mining starts, which is a process by which the blockchain is secured and new coins are generated as a reward for the miners who spend appropriate computational resources. This concept is explained in more detail later in this chapter.
6. Once a miner solves the PoW problem it broadcasts the newly mined block to the network. PoW is explained in detail later in this chapter.
7. The nodes verify the block and propagate the block further, and confirmations start to generate.

8. Finally, the confirmations start to appear in the receiver's wallet and after approximately three confirmations, the transaction is considered finalized and confirmed. However, three to six is just a recommended number; the transaction can be considered final even after the first confirmation. The key idea behind waiting for six confirmations is that the probability of double spending is virtually eliminated after three confirmations.

Transaction fee

Transaction fees are charged by the miners. The fee charged is dependent upon the size and weight of the transaction. Transaction fees are calculated by subtracting the sum of the inputs and the sum of the outputs.

A simple formula can be used:

$$fee = sum(inputs) - sum(outputs)$$

The fees are used as an incentive for miners to encourage them to include a user transaction in the block the miners are creating. All transactions end up in the memory pool, from where miners pick up transactions based on their priority to include them in the proposed block. The calculation of priority is introduced later in this chapter; however, from a transaction fee point of view, a transaction with a higher fee will be picked up sooner by the miners.

There are different rules based on which fee is calculated for various types of actions, such as sending transactions, inclusion in blocks, and relaying by nodes. Fees are not fixed by the Bitcoin protocol and are not mandatory; even a transaction with no fee will be processed in due course but may take a very long time. This is however no longer practical due to the high volume of transactions and competing investors on the Bitcoin network, therefore it is advisable to provide a fee always. The time for transaction confirmation usually ranges from 10 minutes to over 12 hours in some cases. Transaction time is dependent on transaction fees and network activity. If the network is very busy then naturally transactions will take longer to process and if you pay a higher fee then your transaction is more likely to be picked by miners first due to additional incentive of the higher fee.

Transaction pools

Also known as memory pools, these pools are basically created in local memory (computer RAM) by nodes in order to maintain a temporary list of transactions that are not yet confirmed in a block. Transactions are included in a block after passing verification and based on their priority.

The transaction data structure

A transaction at a high level contains metadata, inputs, and outputs. Transactions are combined to create a block.

The transaction data structure is shown in the following table:

Field	**Size**	**Description**
Version number	4 bytes	Used to specify rules to be used by the miners and nodes for transaction processing.
Input counter	1-9 bytes	The number (positive integer) of inputs included in the transaction.
List of inputs	Variable	Each input is composed of several fields, including `Previous Tx hash`, `Previous Txout-index`, `Txin-script length`, `Txin-script`, and optional sequence number. The first transaction in a block is also called a coinbase transaction. It specifies one or more transaction inputs.
Output counter	1-9 bytes	A positive integer representing the number of outputs.
List of outputs	Variable	Outputs included in the transaction.
Lock time	4 bytes	This field defines the earliest time when a transaction becomes valid. It is either a Unix timestamp or block height.

A sample transaction is shown as follows. This is the decoded transaction from the first example of a payment transaction provided at the start of this chapter.

```
{
   "lock_time":0,
   "size":226,
   "inputs":[
      {
```

```
         "prev_out":{
            "index":139,
"hash":"40120e43f00ff96e098a9173f14f1371655b3478bc0a558d6dc17a4ab176387d"
         },
"script":"483045022100de6fd8120d9f142a82d5da9389e271caa3a757b01757c8e4fa7af
bf92e74257c02202a78d4fbd52ae9f3a0083760d76f84643cf8ab80f5ef971e3f98ccba2c71
758d012102c16942555f5e633645895c9affcb994ea7910097b7734a6c2d25468622f25e12"
      }
   ],
   "version":1,
   "vin_sz":1,
"hash":"d28ca5a59b2239864eac1c96d3fd1c23b747f0ded8f5af0161bae8a616b56a1d",
   "vout_sz":2,
   "out":[
      {
         "script_string":"OP_DUP OP_HASH160
c568ffeb46c6a9362e44a5a49deaa6eab05a619a OP_EQUALVERIFY OP_CHECKSIG",
         "address":"1JzouJCVmMQBmTcd8K4Y5BP36gEFNn1ZJ3",
         "value":33324,
         "script":"76a914c568ffeb46c6a9362e44a5a49deaa6eab05a619a88ac"
      },
      {
         "script_string":"OP_DUP OP_HASH160
9386c8c880488e80a6ce8f186f788f3585f74aee OP_EQUALVERIFY OP_CHECKSIG",
         "address":"1ET3oBGf8JpunjytE7owyVtmBjmvcDycQe",
         "value":93376,
         "script":"76a9149386c8c880488e80a6ce8f186f788f3585f74aee88ac"
      }
   ]
}
```

As shown in the preceding code, there are a number of structures that make up the transaction. All these elements are described in the following subsections.

Metadata

This part of the transaction contains some values such as the size of the transaction, the number of inputs and outputs, the hash of the transaction, and a `lock_time` field. Every transaction has a prefix specifying the version number. These fields are shown in the preceding example: `lock_time`, `size`, and `version`.

Inputs

Generally, each input spends a previous output. Each output is considered as **Unspent Transaction Output** (**UTXO**) until an input consumes it. UTXO is an unspent transaction output that can be spent as an input to a new transaction.

Transaction input data structure is shown in the following table:

Field	**Size**	**Description**
Transaction hash	32 bytes	This is the hash of the previous transaction with UTXO.
Output index	4 bytes	This is the previous transactions output index, that is, UTXO to be spent.
Script length	1-9 bytes	This is the size of the unlocking script.
Unlocking script	Variable	Input script (`ScriptSig`) which satisfies the requirements of the locking script.
Sequence number	4 bytes	Usually disabled or contains lock time. Disabled is represented by `'0xFFFFFFFF'`.

In the preceding example the inputs are defined under `"inputs" : [` section.

Outputs

Outputs have three fields, and they contain instructions for sending bitcoins. The first field contains the amount of Satoshis whereas the second field contains the size of the locking script. Finally, the third field contains a locking script that holds the conditions that need to be met in order for the output to be spent. More information on transaction spending using locking and unlocking scripts and producing outputs is discussed later in this section.

Transaction output data structure is shown here:

Field	**Size**	**Description**
Value	8 bytes	Total number in positive integers of Satoshis to be transferred
Script size	1-9 bytes	Size of the locking script
Locking script	Variable	Output script (`ScriptPubKey`)

In the preceding example two outputs are shown under `"OUT":[` section.

Verification

Verification is performed using Bitcoin's scripting language which is described in the next section in detail.

The script language

Bitcoin uses a simple stack-based language called **script** to describe how bitcoins can be spent and transferred. It is not Turing complete and has no loops to avoid any undesirable effects of long-running/hung scripts on the Bitcoin network. This scripting language is based on a Forth programming language like syntax and uses a reverse polish notation in which every operand is followed by its operators. It is evaluated from the left to the right using a **Last In, First Out** (**LIFO**) stack.

Scripts use various opcodes or instructions to define their operation. Opcodes are also known as words, commands, or functions. Earlier versions of the Bitcoin node had a few opcodes that are no longer used due to bugs discovered in their design.

The various categories of the scripting opcodes are constants, flow control, stack, bitwise logic, splice, arithmetic, cryptography, and lock time.

A transaction script is evaluated by combining `ScriptSig` and `ScriptPubKey`. `ScriptSig` is the unlocking script, whereas `ScriptPubKey` is the locking script. This is how a transaction to be spent is evaluated:

1. First, it is unlocked and then it is spent
2. `ScriptSig` is provided by the user who wishes to unlock the transaction
3. `ScriptPubkey` is part of the transaction output and specifies the conditions that need to be fulfilled in order to spend the output
4. In other words, outputs are locked by `ScriptPubKey` that contains the conditions, when met will unlock the output, and coins can then be redeemed

Commonly used opcodes

All opcodes are declared in the `script.h` file in the Bitcoin reference client source code.

This can be accessed from the link at `https://github.com/bitcoin/bitcoin/blob/master/src/script/script.h` under the following comment:
`/** Script opcodes */`

A description of the most commonly used opcodes is listed here. This table is taken from the Bitcoin developer's guide:

Opcode	Description
`OP_CHECKSIG`	This takes a public key and signature and validates the signature of the hash of the transaction. If it matches, then `TRUE` is pushed onto the stack; otherwise, `FALSE` is pushed.
`OP_EQUAL`	This returns `1` if the inputs are exactly equal; otherwise, `0` is returned.
`OP_DUP`	This duplicates the top item in the stack.
`OP_HASH160`	The input is hashed twice, first with SHA-256 and then with RIPEMD-160.
`OP_VERIFY`	This marks the transaction as invalid if the top stack value is not true.
`OP_EQUALVERIFY`	This is the same as `OP_EQUAL`, but it runs `OP_VERIFY` afterwards.
`OP_CHECKMULTISIG`	This takes the first signature and compares it against each public key until a match is found and repeats this process until all signatures are checked. If all signatures turn out to be valid, then a value of `1` is returned as a result; otherwise, `0` is returned.

Types of transactions

There are various scripts available in Bitcoin to handle the value transfer from the source to the destination. These scripts range from very simple to quite complex depending upon the requirements of the transaction. Standard transaction types are discussed here. Standard transactions are evaluated using `IsStandard()` and `IsStandardTx()` tests and only standard transactions that pass the test are generally allowed to be mined or broadcasted on the Bitcoin network. However, nonstandard transactions are valid and allowed on the network.

The following are the standard transaction types:

- **Pay to Public Key Hash** (**P2PKH**): P2PKH is the most commonly used transaction type and is used to send transactions to the bitcoin addresses. The format of the transaction is shown as follows:

```
      ScriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY
OP_CHECKSIG
      ScriptSig: <sig> <pubKey>
```

  The `ScriptPubKey` and `ScriptSig` parameters are concatenated together and executed. An example will follow shortly in this section, where this is explained in more detail.

- **Pay to Script Hash** (**P2SH**): P2SH is used in order to send transactions to a script hash (that is, the addresses starting with `3`) and was standardized in BIP16. In addition to passing the script, the redeem script is also evaluated and must be valid. The template is shown as follows:

```
ScriptPubKey: OP_HASH160 <redeemScriptHash> OP_EQUAL
ScriptSig: [<sig>...<sign>] <redeemScript>
```

- **MultiSig** (**Pay to MultiSig**): M-of-N MultiSig transaction script is a complex type of script where it is possible to construct a script that required multiple signatures to be valid in order to redeem a transaction. Various complex transactions such as escrow and deposits can be built using this script. The template is shown here:

```
ScriptPubKey: <m> <pubKey> [<pubKey> . . . ] <n> OP_CHECKMULTISIG
ScriptSig: 0 [<sig > . . . <sign>]
```

  Raw multisig is obsolete, and multisig is usually part of the P2SH redeem script, mentioned in the previous bullet point.

- **Pay to Pubkey**: This script is a very simple script that is commonly used in coinbase transactions. It is now obsolete and was used in an old version of bitcoin. The public key is stored within the script in this case, and the unlocking script is required to sign the transaction with the private key.

  The template is shown as follows:

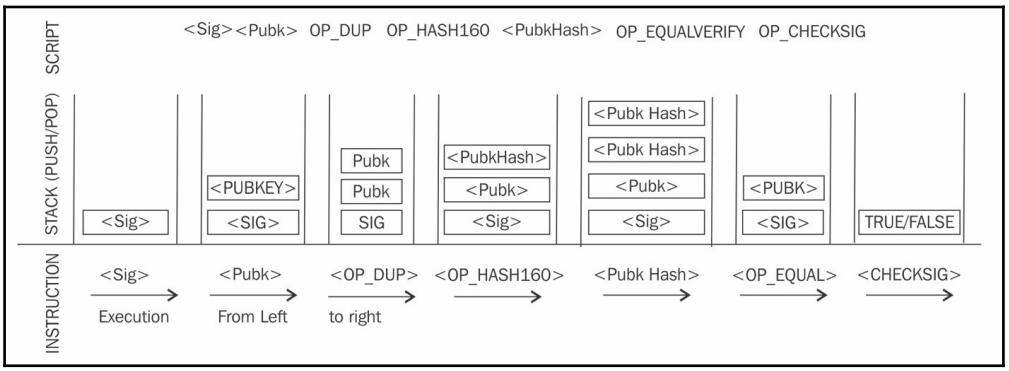
```
<PubKey> OP_CHECKSIG
```

- **Null data/OP_RETURN**: This script is used to store arbitrary data on the blockchain for a fee. The limit of the message is 40 bytes. The output of this script is unredeemable because `OP_RETURN` will fail the validation in any case. `ScriptSig` is not required in this case.

The template is very simple and is shown as follows:

```
OP_RETURN <data>
```

A P2PKH script execution is shown in the following diagram:



P2PKH script execution

All transactions are eventually encoded into the hexadecimal format before transmitting over the Bitcoin network. A sample transaction is shown here in hexadecimal format that is retrieved using `bitcoin-cli` on the Bitcoin node running on mainnet:

```
$ bitcoin-cli getrawtransaction
"d28ca5a59b2239864eac1c96d3fd1c23b747f0ded8f5af0161bae8a616b56a1d"
{
  "result":
"01000000017d3876b14a7ac16d8d550abc78345b6571134ff173918a096ef90ff0430e1240
8b0000006b483045022100de6fd8120d9f142a82d5da9389e271caa3a757b01757c8e4fa7af
bf92e74257c02202a78d4fbd52ae9f3a0083760d76f84643cf8ab80f5ef971e3f98ccba2c71
758d012102c16942555f5e633645895c9affcb994ea7910097b7734a6c2d25468622f25e12f
ffffff022c82000000000001976a914c568ffeb46c6a9362e44a5a49deaa6eab05a619a88
acc06c0100000000001976a9149386c8c880488e80a6ce8f186f788f3585f74aee88ac00000
000",
  "error": null,
  "id": null
}
```

Note that this is the same transaction that was presented as an example at the start of this chapter.

Coinbase transactions

A coinbase transaction or generation transaction is always created by a miner and is the first transaction in a block. It is used to create new coins. It includes a special field, also called `coinbase`, which acts as an input to the coinbase transaction. This transaction also allows up to 100 bytes of arbitrary data that can be used to store arbitrary data. In the genesis block, this transaction included the most famous comment taken from *The Times* newspaper:

> *"The Times 03/Jan/2009 Chancellor on brink of second bailout for banks."*

This message is a proof that the genesis block was not mined earlier than January 3, 2009. This is because first Bitcoin block (genesis block) was created on January 3, 2009 and this news excerpt was taken from that day's newspaper.

A coinbase transaction input has the same number of fields as usual transaction input, but the structure contains coinbase data size and coinbase data fields instead of unlocking script size and unlocking script fields. Also, it does not have a reference pointer to the previous transaction. This structure is shown in the following table:

Field	**Size**	**Description**
Transaction hash	32 bytes	Set to all zeroes as no hash reference is used
Output index	4 bytes	Set to `0xFFFFFFFF`
Coinbase data length	1-9 bytes	2 bytes-100 bytes
Data	Variable	Any data
Sequence number	4 bytes	Set to `0xFFFFFFFF`

Contracts

As defined in the Bitcoin core developer guide, contracts are basically transactions that use the Bitcoin system to enforce a financial agreement. This is a simple definition but has far-reaching consequences as it allows users to design complex contracts that can be used in many real-world scenarios. Contracts allow the development of a completely decentralized, independent, and reduced risk platform.

Various contracts, such as escrow, arbitration, and micropayment channels, can be built using the Bitcoin scripting language. The current implementation of a script is very limited, but various types of contracts are still possible to develop. For example, the release of funds only if multiple parties sign the transaction or perhaps the release of funds only after a certain time has elapsed. Both of these scenarios can be realized using multisig and transaction lock time options.

Transaction verification

This verification process is performed by Bitcoin nodes. The following is described in the Bitcoin developer guide:

1. Check the syntax and ensure that the syntax and data structure of the transaction conforms to the rules provided by the protocol.
2. Verify that no transaction inputs and outputs are empty.
3. Check whether the size in bytes is less than the maximum block size.
4. The output value must be in the allowed money range (0 to 21 million BTC).
5. All inputs must have a specified previous output, except for coinbase transactions, which should not be relayed.
6. Verify that `nLockTime` must not exceed 31-bits. (`nLockTime` specifies the time before which transaction will not be included in the block.)
7. For a transaction to be valid, it should not be less than 100 bytes.
8. The number of signature operations in a standard transaction should be less than or not more than two.
9. Reject nonstandard transactions; for example, `ScriptSig` is allowed to only push numbers on the stack. `ScriptPubkey` not passing the `isStandard()` checks. The `isStandard()` checks specify that only standard transactions are allowed.
10. A transaction is rejected if there is already a matching transaction in the pool or in a block in the main branch.
11. The transaction will be rejected if the referenced output for each input exists in any other transaction in the pool.
12. For each input, there must exist a referenced output unspent transaction.
13. For each input, if the referenced output transaction is the coinbase, it must have at least 100 confirmations; otherwise, the transaction will be rejected.
14. For each input, if the referenced output does not exist or has been spent already, the transaction will be rejected.

15. Using the referenced output transactions to get input values, verify that each input value, as well as the sum, is in the allowed range of 0-21 million BTC. Reject the transaction if the sum of input values is less than the sum of output values.
16. Reject the transaction if the transaction fee would be too low to get into an empty block.
17. Each input unlocking script must have corresponding valid output scripts.

Transaction malleability

Transaction malleability in Bitcoin was introduced due to a bug in the bitcoin implementation. Due to this bug, it became possible for an adversary to change the transaction ID of a transaction, thus resulting in a scenario where it would appear that a certain transaction has not been executed. This can allow scenarios where double deposits or withdrawals can occur. In other words, this bug allows the changing of the unique ID of a Bitcoin transaction before it is confirmed. If the ID is changed before confirmation, it would seem that the transaction did not occur at all which can then allow these attacks.

Blockchain

Blockchain is a public ledger of a timestamped, ordered, and immutable list of all transactions on the Bitcoin network. Each block is identified by a hash in the chain and is linked to its previous block by referencing the previous block's hash.

In the following table structure of a block is presented, followed by a detailed diagram that provides a detailed view of the blockchain structure.

The structure of a block

The following table shows the structure of a block:

Field	Size	Description
Block size	4 bytes	This is the size of the block.
Block header	80 bytes	This includes fields from the block header described in the next section.

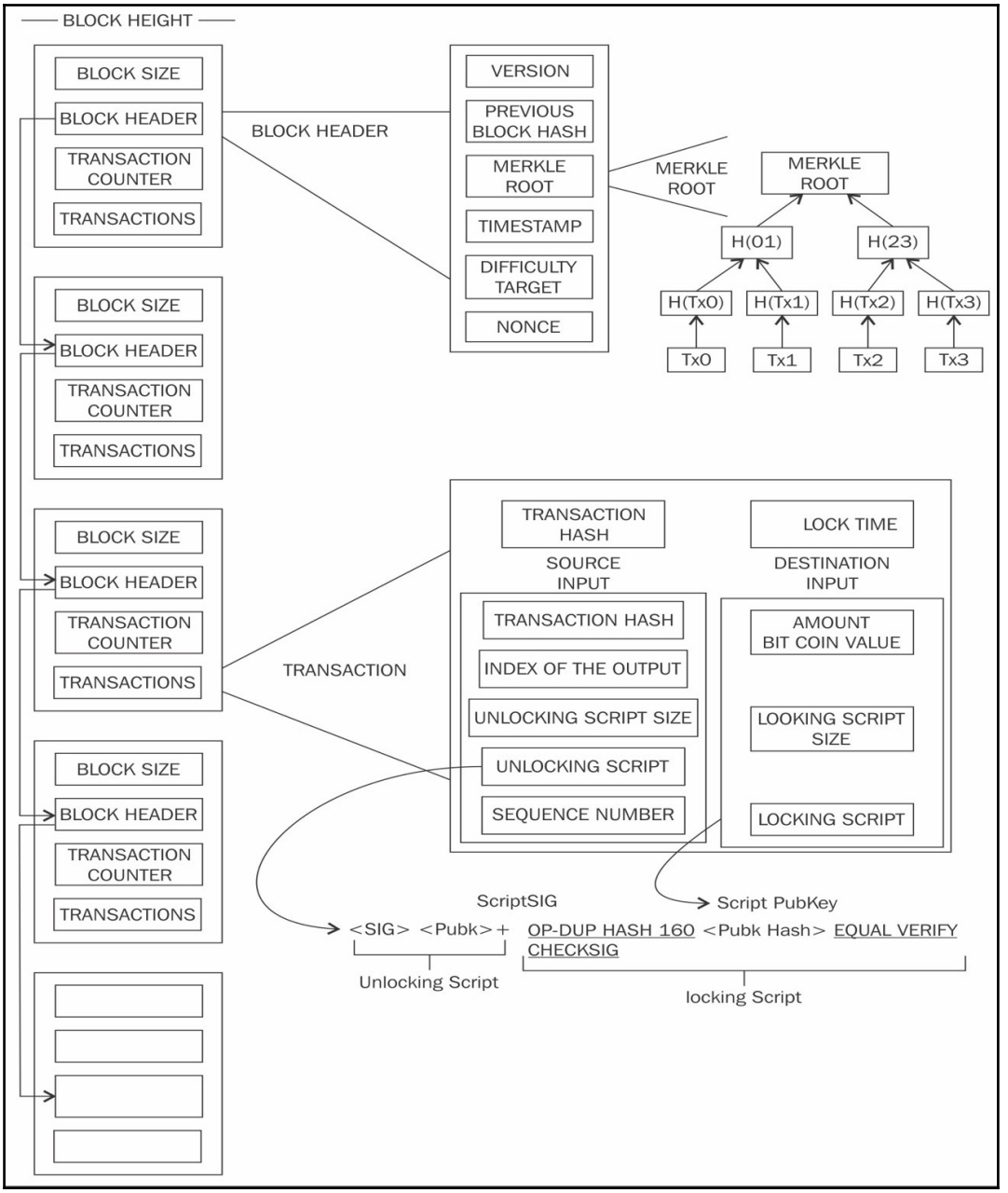
Transaction counter	Variable	This field contains the total number of transactions in the block, including the coinbase transaction. Size ranges from 1-9 bytes
Transactions	Variable	All transactions in the block.

The structure of a block header

The following table depicts the structure of a block header:

Field	**Size**	**Description**
Version	4 bytes	The block version number that dictates the block validation rules to follow.
Previous block's header hash	32 bytes	This is a double SHA-256 hash of the previous block's header.
Merkle root hash	32 bytes	This is a double SHA-256 hash of the Merkle tree of all transactions included in the block.
Timestamp	4 bytes	This field contains the approximate creation time of the block in the Unix epoch time format. More precisely, this is the time when the miner has started hashing the header. (The time from the miner's point of view.)
Difficulty target	4 bytes	This is the current difficulty target of the network/block.
Nonce	4 bytes	This is an arbitrary number that miners change repeatedly to produce a hash that is lower than the difficulty target.

As shown in the following diagram, blockchain is a chain of blocks where each block is linked to its previous block by referencing the previous block header's hash. This linking makes sure that no transaction can be modified unless the block that records it and all blocks that follow it are also modified. The first block is not linked to any previous block and is known as the genesis block.



A visualization of the blockchain, block, block header, transactions and scripts

The preceding diagram shows a high-level overview of the Bitcoin blockchain. On the left-hand side blocks are shown starting from top to bottom. Each block contains transactions and block headers which are further magnified on the right-hand side. On the top, first, block header is expanded to show various elements within the block header. Then on the right-hand side the Merkle root element of the block header is shown in magnified view which shows that how Merkle root is calculated. We have discussed Merkle trees in detail previously, you can refer to `Chapter 3`, *Symmetric Cryptography* if you need to revise the concept. Further down transactions are also magnified to show the structure of a transaction and the elements that it contains. Also, note that transactions are then further elaborated by showing that what locking and unlocking scripts look like. This diagram shows a lot of components, we will discuss all these in this chapter.

The genesis block

This is the first block in the Bitcoin blockchain. The genesis block was hardcoded in the bitcoin core software. It is in the `chainparams.cpp` file (`https://github.com/bitcoin/bitcoin/blob/master/src/chainparams.cpp`):

```
static CBlock CreateGenesisBlock(uint32_t nTime, uint32_t nNonce, uint32_t
nBits, int32_t nVersion, const CAmount& genesisReward)
{
   const char* pszTimestamp = "The Times 03/Jan/2009 Chancellor on brink of
second bailout for banks";
   const CScript genesisOutputScript = CScript() <<
ParseHex("04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61deb
649f6bc3f4cef38c4f35504e51ec112de5c384df7ba0b8d578a4c702b6bf11d5f") <<
OP_CHECKSIG;
   return CreateGenesisBlock(pszTimestamp, genesisOutputScript, nTime,
nNonce,
   nBits, nVersion, genesisReward);
}
```

Bitcoin provides protection against double spending by enforcing strict rules on transaction verification and via mining. Transactions and blocks are added in the blockchain only after strict rule checking explained earlier in the *Transaction verification* section and successful PoW solution. Block height is the number of blocks before a particular block in the blockchain. The current height (as of March 6, 2018) of the blockchain is 512,328 blocks. PoW is used to secure the blockchain. Each block contains one or more transactions, out of which the first transaction is a coinbase transaction. There is a special condition for coinbase transactions that prevent them from being spent until at least 100 blocks in order to avoid a situation where the block may be declared stale later on.

Stale blocks are created when a block is solved and every other miner who is still working to find a solution to the hash puzzle is working on that block. Mining and hash puzzles will be discussed later in the chapter in detail. As the block is no longer required to be worked on, this is considered a stale block.

Orphan blocks are also called detached blocks and were accepted at one point in time by the network as valid blocks but were rejected when a proven longer chain was created that did not include this initially accepted block. They are not part of the main chain and can occur at times when two miners manage to produce the blocks at the same time.

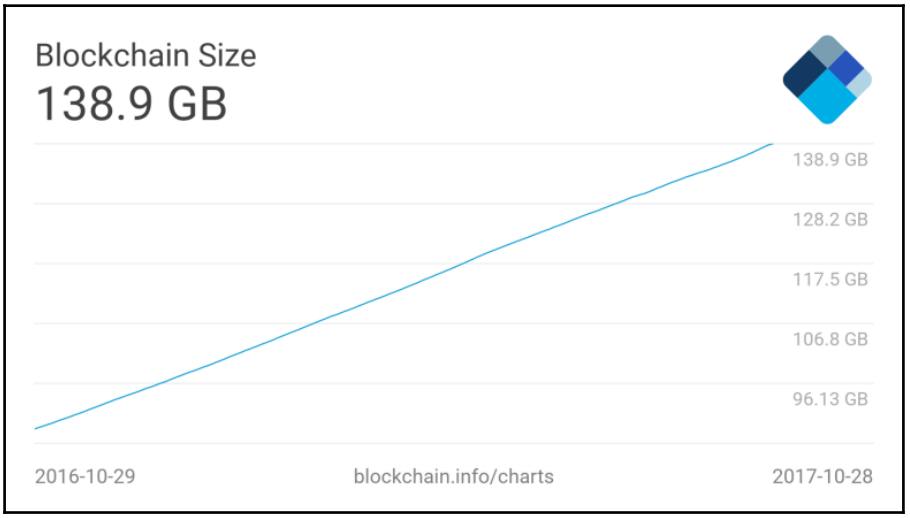
The latest block version is version 4, which was proposed with BIP65 and has been used since bitcoin core client 0.11.2 since the implementation of BIP9 bits in the `nVersion` field are being used to indicate soft fork changes.

Because of the distributed nature of bitcoin, network forks can occur naturally. In cases where two nodes simultaneously announce a valid block can result in a situation where there are two blockchains with different transactions. This is an undesirable situation but can be addressed by the Bitcoin network only by accepting the longest chain. In this case, the smaller chain will be considered orphaned. If an adversary manages to gain 51% control of the network hash rate (computational power), then they can impose their own version of transaction history.

Forks in blockchain can also occur with the introduction of changes in the Bitcoin protocol. In case of a *soft fork*, a client which chooses not to upgrade to the latest version supporting the updated protocol will still be able to work and operate normally. In this case, previous and new blocks are both acceptable, thus making soft fork backwards compatible.

In case of a soft fork, only miners are required to upgrade to the new client software in order to make use of the new protocol rules. Planned upgrades do not necessarily create forks because all users should have updated already. A hard fork, on the other hand, invalidates previously valid blocks and requires all users to upgrade. New transaction types are sometimes added as a soft fork, and any changes such as block structure change or major protocol changes results in a hard fork. The current size of the bitcoin blockchain as of October 29, 2017, stands at approximately 139 GB.

The following diagram shows the size increase of blockchain as a function of time:



Current size of blockchain as of 29/10/2017

New blocks are added to the blockchain approximately every 10 minutes and network difficulty is adjusted dynamically every 2016 blocks in order to maintain a steady addition of new blocks to the network.

Network difficulty is calculated using the following equation:

$$Target = Previous\ target * Time/2016 * 10\ minutes$$

Difficulty and target are interchangeable and represent the same thing. Previous target represents the old target value, and time is the time spent to generate previous 2016 blocks. Network difficulty basically means how hard it is for miners to find a new block, that is, how difficult the hashing puzzle is now.

In the next section, mining is discussed, which will explain how the hashing puzzle is solved.

Mining

Mining is a process by which new blocks are added to the blockchain. Blocks contain transactions that are validated via the mining process by mining nodes on the Bitcoin network. Blocks, once mined and verified are added to the blockchain which keeps the blockchain growing. This process is resource-intensive due to the requirements of PoW where miners compete in order to find a number which is less than the difficulty target of the network. This difficulty in finding the correct value (also called sometimes the mathematical puzzle) is there to ensure that the required resources have been spent by miners before a new proposed block can be accepted. New coins are minted by the miners by solving the PoW problem, also known as partial hash inversion problem. This process consumes a high amount of resources including computing power and electricity. This process also secures the system against frauds and double spending attacks while adding more virtual currency to the Bitcoin ecosystem.

Roughly one new block is created (mined) every 10 minutes to control the frequency of generation of bitcoins. This frequency needs to be maintained by the Bitcoin network and is encoded in the bitcoin core clients in order to control the *money supply*. Miners are rewarded with new coins if and when they discover new blocks by solving PoW. Miners are paid transaction fees in return for including transactions in their proposed blocks. New blocks are created at an approximate fixed rate of every 10 minutes. The rate of creation of new bitcoins decreases by 50%, every 210,000 blocks, roughly every 4 years. When bitcoin was initially introduced, the block reward was 50 bitcoins; then in 2012, this was reduced to 25 bitcoins. In July 2016, this was further reduced to 12.5 coins (12 coins) and the next reduction is estimated to be on July 4, 2020. This will reduce the coin reward further down to approximately six coins.

Approximately 144 blocks, that is, 1,728 bitcoins are generated per day. The number of actual coins can vary per day; however, the number of blocks remains at 144 per day. Bitcoin supply is also limited and in 2140, almost 21 million bitcoins will be finally created and no new bitcoins can be created after that. Bitcoin miners, however, will still be able to profit from the ecosystem by charging transaction fees.

Tasks of the miners

Once a node connects to the bitcoin network, there are several tasks that a bitcoin miner performs:

1. **Synching up with the network**: Once a new node joins the bitcoin network, it downloads the blockchain by requesting historical blocks from other nodes. This is mentioned here in the context of the bitcoin miner; however, this not necessarily a task only for a miner.
2. **Transaction validation**: Transactions broadcasted on the network are validated by full nodes by verifying and validating signatures and outputs.
3. **Block validation**: Miners and full nodes can start validating blocks received by them by evaluating them against certain rules. This includes the verification of each transaction in the block along with verification of the nonce value.
4. **Create a new block**: Miners propose a new block by combining transactions broadcasted on the network after validating them.
5. **Perform Proof of Work**: This task is the core of the mining process and this is where miners find a valid block by solving a computational puzzle. The block header contains a 32-bit nonce field and miners are required to repeatedly vary the nonce until the resultant hash is less than a predetermined target.
6. **Fetch reward**: Once a node solves the hash puzzle (PoW), it immediately broadcasts the results, and other nodes verify it and accept the block. There is a slight chance that the newly minted block will not be accepted by other miners on the network due to a clash with another block found at roughly the same time, but once accepted, the miner is rewarded with 12.5 bitcoins and any associated transaction fees.

Mining rewards

When Bitcoin started in 2009 the mining reward used to be 50 bitcoins. After every 210,000 blocks, the block reward halves. In November 2012 it halved down to 25 bitcoins. Currently, it is 12.5 BTC per block since July 2016. Next halving is on Friday, 12 June 2020 after which the block reward will be reduced down to 6.25 BTC per block. This mechanism is hardcoded in Bitcoin to regulate, control inflation and limit the supply of bitcoins.

Proof of Work (PoW)

This is a proof that enough computational resources have been spent in order to build a valid block. PoW is based on the idea that a random node is selected every time to create a new block. In this model, nodes compete with each other in order to be selected in proportion to their computing capacity. The following equation sums up the PoW requirement in bitcoin:

$$H (N \mid\mid P\_hash \mid\mid Tx \mid\mid Tx \mid\mid \ldots Tx) < Target$$

Where *N* is a nonce, *P_hash* is a hash of the previous block, *Tx* represents transactions in the block, and *Target* is the target network difficulty value. This means that the hash of the previously mentioned concatenated fields should be less than the target hash value.

The only way to find this nonce is the brute force method. Once a certain pattern of a certain number of zeroes is met by a miner, the block is immediately broadcasted and accepted by other miners.

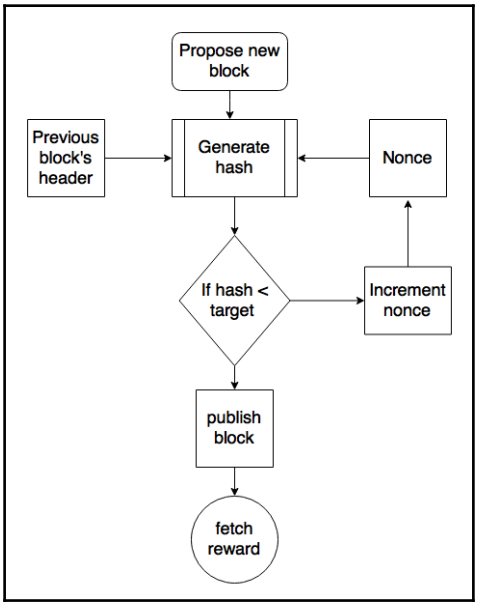
The mining algorithm

The mining algorithm consists of the following steps.

1. The previous block's header is retrieved from the bitcoin network.
2. Assemble a set of transactions broadcasted on the network into a block to be proposed.
3. Compute the double hash of the previous block's header combined with a nonce and the newly proposed block using the SHA-256 algorithm.
4. Check if the resultant hash is lower than the current difficulty level (target) then PoW is solved. As a result of successful PoW the discovered block is broadcasted to the network and miners fetch the reward.
5. If the resultant hash is not less than the current difficulty level (target), then repeat the process after incrementing the nonce.

   As the hash rate of the bitcoin network increased, the total amount of 32-bit nonce was exhausted too quickly. In order to address this issue, the extra nonce solution was implemented, whereby the coinbase transaction is used as a source of extra nonce to provide a larger range of nonce to be searched by the miners.

This process can be visualized by using the following flowchart:



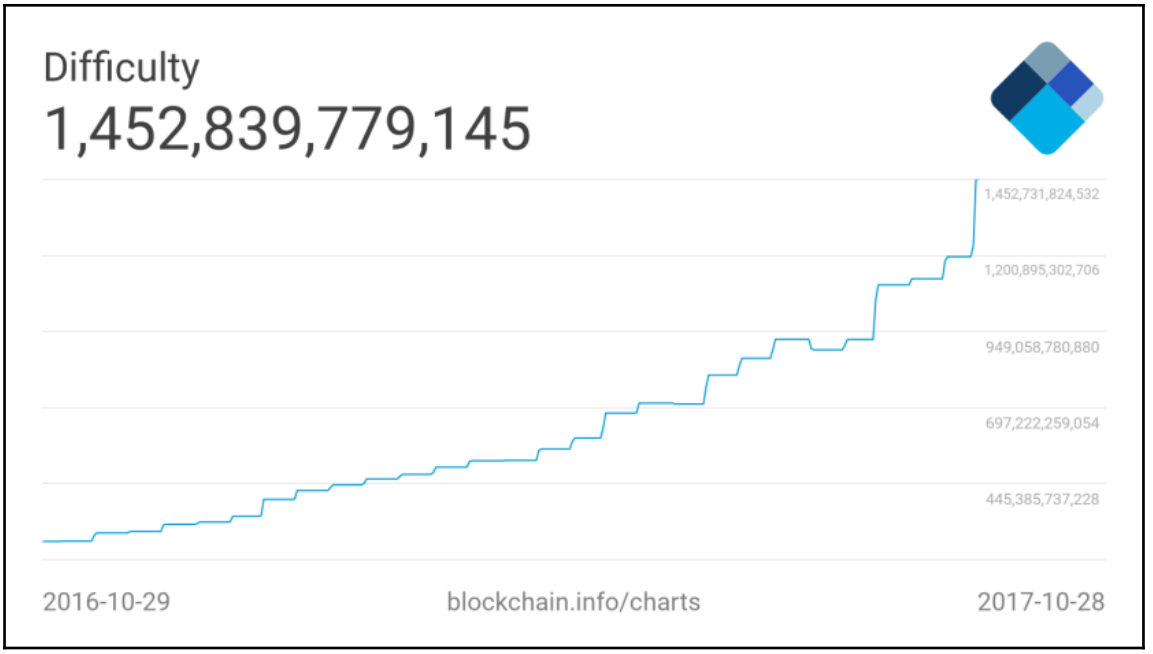
Mining process

Mining difficulty increased over time and bitcoins that could be mined by single CPU laptop computers now require dedicated mining centers to solve the hash puzzle. The current difficulty level can be queried using the Bitcoin command-line interface using the following command:

```
$ bitcoin-cli getdifficulty
1452839779145
```

This number represents the difficulty level of the Bitcoin network. Recall from previous sections that miners compete to find a solution to a problem. This number, in fact shows, that how difficult it is to find a hash which is lower than the network difficulty target. All successfully mined blocks must contain a hash that is less than this target number. This number is updated every 2 weeks or 2016 blocks to ensure that on average 10-minute block generation time is maintained.

Bitcoin network difficulty has increased exponentially, the following graph shows this difficulty level over a period of one year:

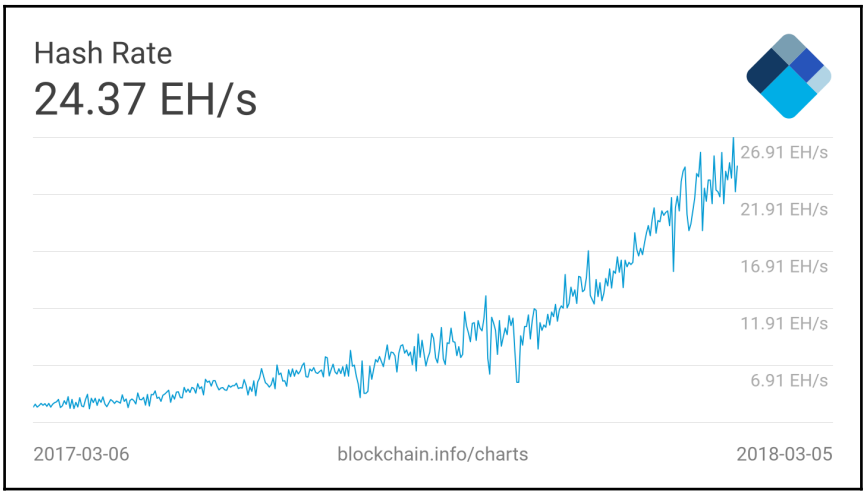


Mining difficulty over the last year

The preceding graph shows the difficulty of the Bitcoin network over a period of last year and it has increased quite significantly. The reason why mining difficulty increases is because in Bitcoin, the block generation time has to be always around 10 minutes. This means that if blocks are being mined too quickly by fast hardware then the difficulty increases so that the block generation time can remain at roughly 10 minutes per block. This is also true in reverse if blocks are not mined every 10 minutes than the difficulty is decreased. Difficulty, is calculated every 2016 blocks (in two weeks) and adjusted accordingly. If the previous set of 2016 blocks were mined in less than a period of two weeks then difficulty will be increased. Similarly, if 2016 blocks were found in more than two weeks (If blocks are mined every 10 minutes then 2016 blocks take 2 weeks to be mined) then the difficulty is decreased.

The hash rate

The hashing rate basically represents the rate of calculating hashes per second. In other words, this is the speed at which miners in the Bitcoin network are calculating hashes to find a block. In early days of bitcoin, it used to be quite small as CPUs were used. However, with dedicated mining pools and ASICs now, this has gone up exponentially in the last few years. This has resulted in increased difficulty of the Bitcoin network. The following hash rate graph shows the hash rate increase over time and is currently measured in Exa hashes. This means that in 1 second, the Bitcoin network miners are computing more than 24,000,000,000,000,000,000 hashes per second.



Hashing rate (measured in Exa-hashes) as of March 2018, shown over a period of 1 year

Mining systems

Over time, bitcoin miners have used various methods to mine bitcoins. As the core principle behind mining is based on the double SHA-256 algorithm, overtime experts have developed sophisticated systems to calculate the hash faster and faster. The following is a review of the different types of mining methods used in bitcoin and how they evolved with time.

CPU

CPU mining was the first type of mining available in the original bitcoin client. Users could even use laptop or desktop computers to mine bitcoins. CPU mining is no longer profitable and now more advanced mining methods such as ASIC-based mining is used. CPU mining only lasted for around just over a year since the introduction of Bitcoin and soon other methods were explored and tried by the miners.

GPU

Due to the increased difficulty of the bitcoin network and the general tendency of finding faster methods to mine, miners started to use GPUs or graphics cards available in PCs to perform mining. GPUs support faster and parallelized calculations that are usually programmed using the OpenCL language. This turned out to be a faster option as compared to CPUs. Users also used techniques such as overclocking to gain maximum benefit of the GPU power. Also, the possibility of using multiple graphics cards increased the popularity of graphics cards' usage for bitcoin mining. GPU mining, however, has some limitations, such as overheating and the requirement for specialized motherboards and extra hardware to house multiple graphics cards. From another angle, graphics cards have become quite expensive due to increased demand and this has impacted gamers and graphic software users.

FPGA

Even GPU mining did not last long, and soon miners found another way to perform mining using FPGAs. **Field Programmable Gate Array** (**FPGA**) is basically an integrated circuit that can be programmed to perform specific operations. FPGAs are usually programmed in **Hardware Description Languages** (**HDLs**), such as Verilog and VHDL. Double SHA-256 quickly became an attractive programming task for FPGA programmers and several open source projects started too. FPGA offered much better performance as compared to GPUs; however, issues such as accessibility, programming difficulty, and the requirement for specialized knowledge to program and configure FPGAs resulted in a short life of the FPGA era for bitcoin mining.

The arrival of ASICs resulted in quickly phased out FPGA- based systems for mining. Mining hardware such as X6500 miner, Ztex, and Icarus were developed during the time when FPGA mining was profitable. Various FPGA manufacturers, such as Xilinx and Altera, produce FPGA hardware and development boards that can be used to program mining algorithms. It should be noted that GPU mining is still profitable for some other cryptocurrencies to some extent such as Zcoin (`https://zcoin.io/guide-on-how-to-mine-zcoin-xzc/`), but not Bitcoin, because network difficulty of Bitcoin is so high that only ASICs (specialized hardware) running in parallel can produce some reasonable profit.

ASICs

Application Specific Integrated Circuit (**ASIC**) was designed to perform the SHA-256 operation. These special chips were sold by various manufacturers and offered a very high hashing rate. This worked for some time, but due to the quickly increasing mining difficulty level, single-unit ASICs are no longer profitable.

Currently, mining is out of the reach of individuals as vast amounts of energy and money is needed to be spent in order to build a profitable mining platform. Now professional mining centers using thousands of ASIC units in parallel are offering mining contracts to users to perform mining on their behalf. There is no technical limitation, a single user can run thousands of ASICs in parallel but it will require dedicated data centers and hardware, therefore, cost for a single individual can become prohibitive. The following are the four types of mining hardware:

CPU

GPU

FPGA

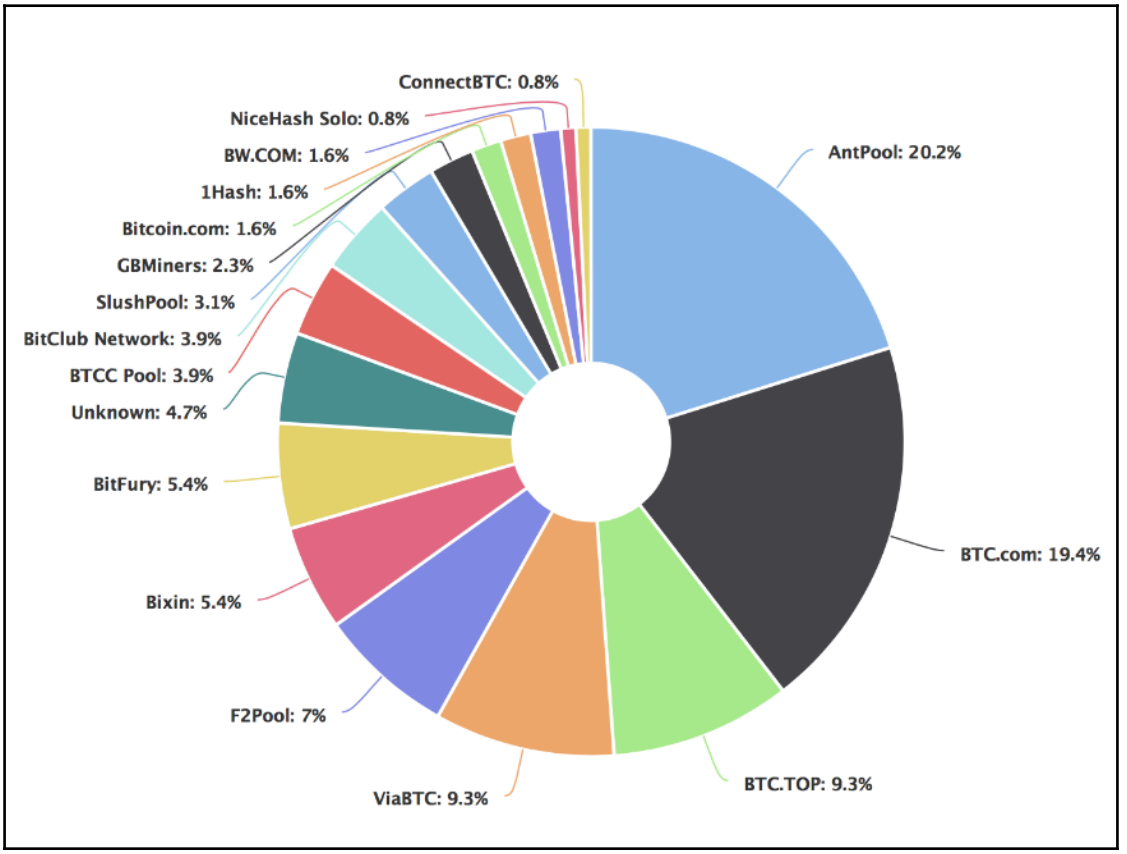
ASIC

Mining pools

A mining pool forms when group of miners work together to mine a block. The pool manager receives the coinbase transaction if the block is successfully mined, which is then responsible for distributing the reward to the group of miners who invested resources to mine the block. This is profitable as compared to solo mining, where only one sole miner is trying to solve the partial hash inversion function (hash puzzle) because, in mining pools, the reward is paid to each member of the pool regardless of whether they (more specifically, their individual node) solved the puzzle or not.

There are various models that a mining pool manager can use to pay to the miners, such as the **Pay Per Share** (**PPS**) model and the proportional model. In the PPS model, the mining pool manager pays a flat fee to all miners who participated in the mining exercise, whereas in the proportional model, the share is calculated based on the amount of computing resources spent to solve the hash puzzle.

Many commercial pools now exist and provide mining service contracts via the cloud and easy-to-use web interfaces. The most commonly used ones are AntPool (`https://www.antpool.com`), BTC (`https://btc.com`), and BTC TOP (`http://www.btc.top`). A comparison of hashing power for all major mining pools is shown in the following diagram:



Mining pools and their hashing power (hash rate) as of 28/10/2017

Source: https://blockchain.info/pools

Mining centralization can occur if a pool manages to control more than 51% of the network by generating more than 51% hash rate of the Bitcoin network.

As discussed earlier in the introduction section, a 51% attack can result in successful double-spending attacks, and it can impact consensus and in fact impose another version of transaction history on the Bitcoin network.

This event has happened once in the Bitcoin history when GHash.IO, a large mining pool, managed to acquire more than 51% of the network capacity. Theoretical solutions, such as two-phase PoW (`http://hackingdistributed.com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools/`), have been proposed in academia to disincentivize large mining pools. This scheme introduces a second cryptographic puzzle that results in mining pools to either reveal their private keys or provide a considerable portion of the hash rate of their mining pool, thus reducing the overall hash rate of the pool.

Various types of hardware are commercially available for mining purposes. Currently, the most profitable one is ASIC mining, and specialized hardware is available from a number of vendors such as Antminer, AvalonMiner and Whatsminer. Solo mining is not much profitable now unless a vast amount of money and energy is spent to build your own mining rig or even a data center. With the current difficulty factor (March 2018), if a user manages to produce a hash rate of 12 TH/s, they can hope to make 0.0009170 BTC (around $6) per day, which is very low as compared to the investment required to source the equipment that can produce 12 TH. Including running costs such as electricity, this turns out to be not very profitable.

For example, Antminer S9, is an efficient ASIC miner which produces hash power of 13.5 TH/s and it seems that it can produce some profit per day, which is true but a single Antminer S9 costs around 1700 GBP and combining it with electricity cost the return on investment is almost after a year's mining when it produces around 0.3 BTC. It may seem still OK, to invest but also think about the fact that the Bitcoin network difficulty keeps going up with time and during a year it will become more difficult to mine and the mining hardware will run out its utility in a few months.

Summary

We started this chapter by introducing Bitcoin and how a transaction works from a user's point of view. Then, we presented an introduction to transactions from a technical point of view. Later we discussed public and private keys that are used in Bitcoin.

In the following section, we presented addresses and its different types, following it with a discussion on transactions, its types, and usage. Next, we looked at blockchain with a detailed explanation regarding how blockchain works and what various components are included in the Bitcoin blockchain.

In the last few sections of the chapter, we presented the mining process and relevant concepts.

In the next chapter, we will examine concepts related to the Bitcoin network, its elements, and client software tools.