UNIT – III

INTRODUCTION TO BITCOIN:

Bitcoin is the first application of the blockchain technology.Bitcoin has started a revolution with the introduction of the very first fully decentralized digital currency, and one that has proven to be extremely secure and stable. This has also sparked a great interest in academic and industrial research and introduced many new research areas.

Since its introduction in 2008, bitcoin has gained much popularity and is currently the most successful digital currency in the world with billions of dollars invested in it. 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.

Bitcoin: A Peer-to-Peer Electronic Cash System was written by Satoshi Nakamoto. The first key idea introduce was that 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.

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, and software that facilitate the creation and usage of the digital currency named bitcoin. 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. Nodes in this peer-to-peer network talk to each other using the Bitcoin protocol.

Decentralization of currency was made possible for the first time with the invention of bitcoin. Moreover, the

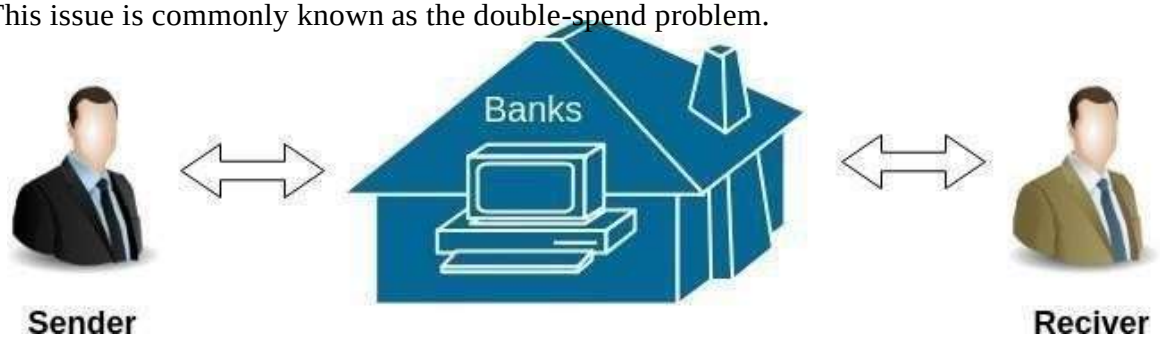
double spending problem was solved in an elegant and ingenious way in bitcoin. Double spending problem arises when, for example, a user sends coins to two different users at the same time and they are verified independently as valid transactions.

Bitcoin Working Mechanism:

When you send an email to another person, you just type an email address and can communicate directly to that person. It is the same thing when you send an instant message. This type of communication between two parties is commonly known as Peer-to-Peer communication.

Whenever you want to transfer money to someone over the internet, you need to use a service of third-party such as banks, a credit card, a PayPal, or some other type of money transfer services. The reason for using third-party is to ensure that you are transferring that money. In other words, you need to be able to verify that both parties have done what they need to do in real exchange.

For example, Suppose you click on a photo that you want to send it to another person, so you can simply attach that photo to an email, type the receiver email address and send it. The other person will receive the photo, and you think it would end, but it is not. Now, we have two copies of photo, one is a simple email, and another is an original file which is still on my computer. Here, we send the copy of the file of the photo, not the original file. This issue is commonly known as the double-spend problem.



The double-spend problem provides a challenge to determine whether a transaction is real or not. How you can send a bitcoin to someone over the internet without needing a bank or some other institution to certify the transfer took place. The answer arises in a global network of thousands of computers called a Bitcoin Network and a special type of decentralized laser technology called **blockchain**.

Transactions & Structure:

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 sends 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 ofthe 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

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 include this transaction in the next block to be mined.
5. Mining starts once a miner who solves the Proof of Work problem broadcasts the newly mined block to the network.
6. The nodes verify the block and propagate the block further, and confirmation starts to generate.
7. Finally, the confirmations start to appear in the receiver's wallet and after approximately six confirmations, the transaction is considered finalized and confirmed. However, 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 six confirmations.

TRANSACTION STRUCTURE

A transaction at a high level contains metadata, inputs, and outputs. Transactions are combined to create a block. The transaction 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 bytes – 9 bytes	The number of inputs included in the transaction.
list of inputs	variable	Each input is composed of several fields, including Previous transaction 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.
Out-counter	1 bytes – 9 bytes	A positive integer representing the number of outputs.
list of outputs	variable	Outputs included in the transaction.
lock_time	4 bytes	This defines the earliest time when a transaction becomes valid. It is either a Unix timestamp or a block number.

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.

Inputs: Generally, each input spends a previous output. Each output is considered an Unspent Transaction Output (UTXO) until an input consumes it.

Outputs: Outputs have only two fields, and they contain instructions for the sending of bitcoins. The first field contains the amount of Satoshis, whereas the second field is a locking script that contains the conditions that need to be met in order for the output to be spent.

Verification: Verification is performed using bitcoin's scripting language.

Types of transaction:

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 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.

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 folows:

ScriptPubKey: OP_DUP OP_HASH160 OP_EQUALVERIFY OP_CHECKSIG ScriptSig: The ScriptPubKey and ScriptSig parameters are concatenated together and executed.

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 OP_EQUAL

ScriptSig: [...]

MultiSig (Pay to MultiSig): M of n multisignature 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: [. . .] OP_CHECKMULTISIG

ScriptSig: 0 [. . .] Raw multisig is obsolete, and multisig is usually part of the P2SH redeem script, mentioned in the previous bullet point.

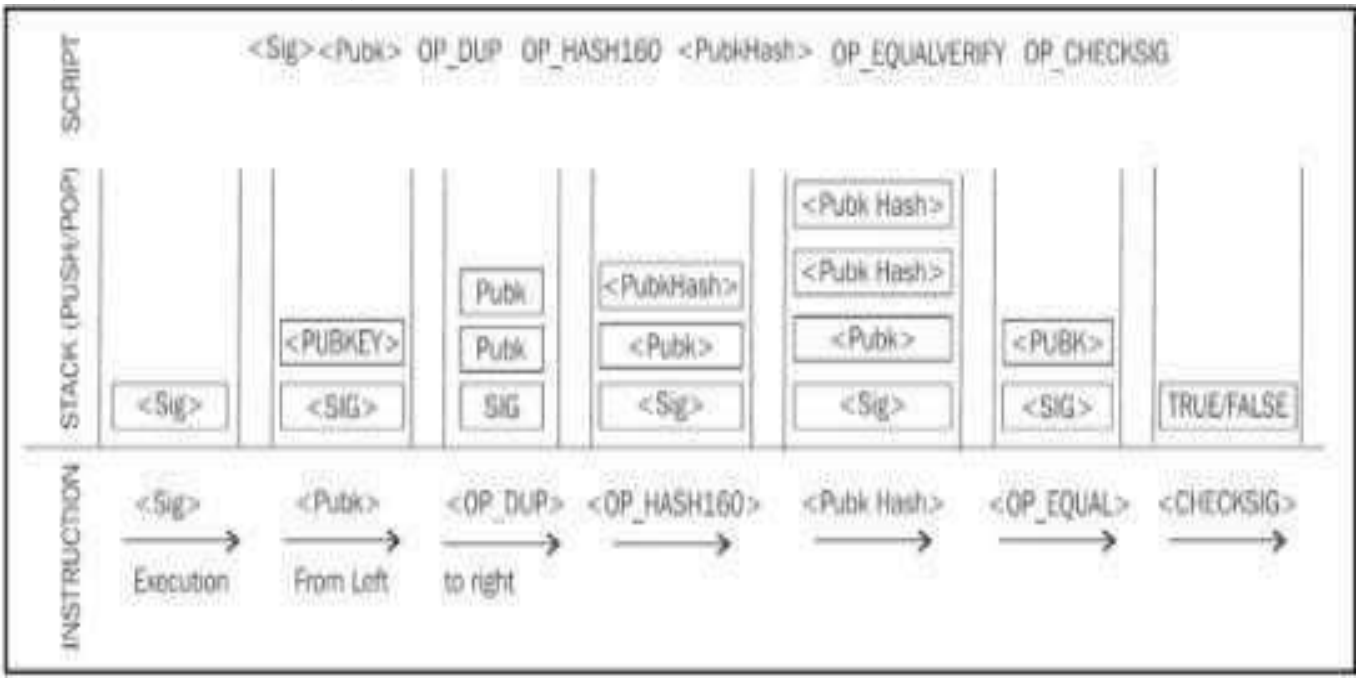
Pay to Pubkey: This script is a very simple script that is commonly used in coinbase transactions. It is now obsolete and was used in an old version of bitcoin. The public key is stored within the script in this case, and the unlocking script is required to sign the transaction with the private key.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

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 as follows:



P2PKH script execution:

All transactions are eventually encoded into the hex before transmitting over the bitcoin network.

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 structure of a block, a block header is described, followed by a detailed diagram that provides an insight into the blockchain structure.

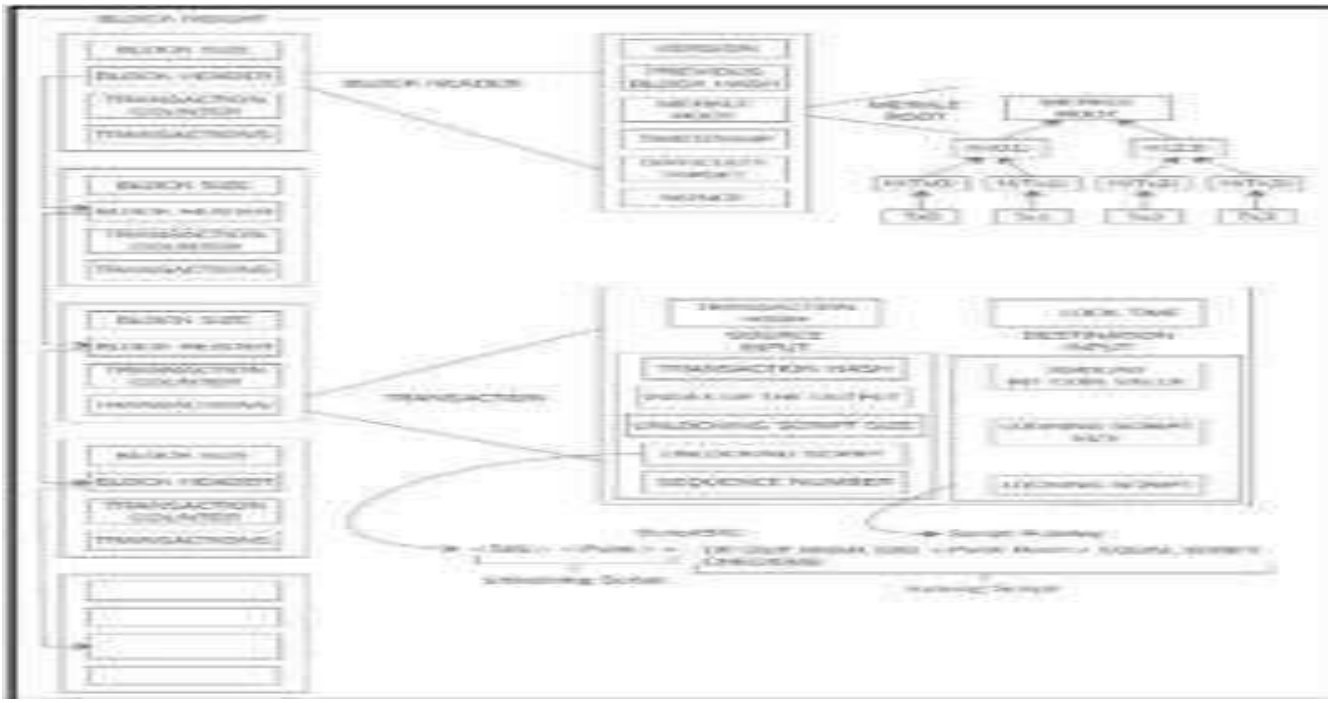
The structure of a block

Bytes	Name	Description
80	Block header	This includes fields from the block header described in the next section.
variable	Transaction counter	The field contains the total number of transactions in the block, including the coinbase transaction.
variable	Transactions	All transactions in the block.

The structure of a block header

Bytes	Name	Description
4	Version	The block version number that dictates the block validation rules to follow.
32	previous block header hash	This is a double SHA256 hash of the previous block's header.
32	merkle root hash	This is a double SHA256 hash of the merkle tree of all transactions included in the block.

4	Timestamp	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).
4	Difficulty target	This is the difficulty target of the block.
4	Nonce	This is an arbitrary number that miners change repeatedly in order to produce a hash that fulfills the difficulty target threshold.



A visualization of blockchain, block, block header, transaction and script.

As shown in the preceding 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 anyprevious block and is known as the genesis block.

The Genesis Block:

This is the first block in the bitcoin blockchain. The genesis block was hardcoded in the bitcoin core software.

Bitcoin provides protection against double spending by enforcing strict rules on transaction verification and via mining. Blocks are added in the blockchain only after strict rule checking and successful Proof of Work solution. Block height is the number of blocks before a particular block in the blockchain. The current height (at the time of writing this) of the blockchain is 434755 blocks. Proof of Work 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 to be 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 Bitcoin Network :

The bitcoin network is a P2P network where nodes exchange transactions and blocks. There are different types of nodes on the network. There are two main types of nodes, full nodes and SPV nodes. Full nodes, as the name implies, are implementations of bitcoin core clients performing the wallet, miner, full blockchain storage, and network routing functions. However, it is not necessary to perform all these functions. SPV nodes or lightweight clients perform only wallet and network routing functionality. The latest version of Bitcoin protocol is 70014 and was introduced with bitcoin core client 0.13.0.

Bitcoin network is identified by its different magic values. A list is shown as follows:

Network	Magic value	Hex
main	0xD9B4BEF9	F9 BE B4 D9
testnet3	0x0709110B	0B 11 09 07

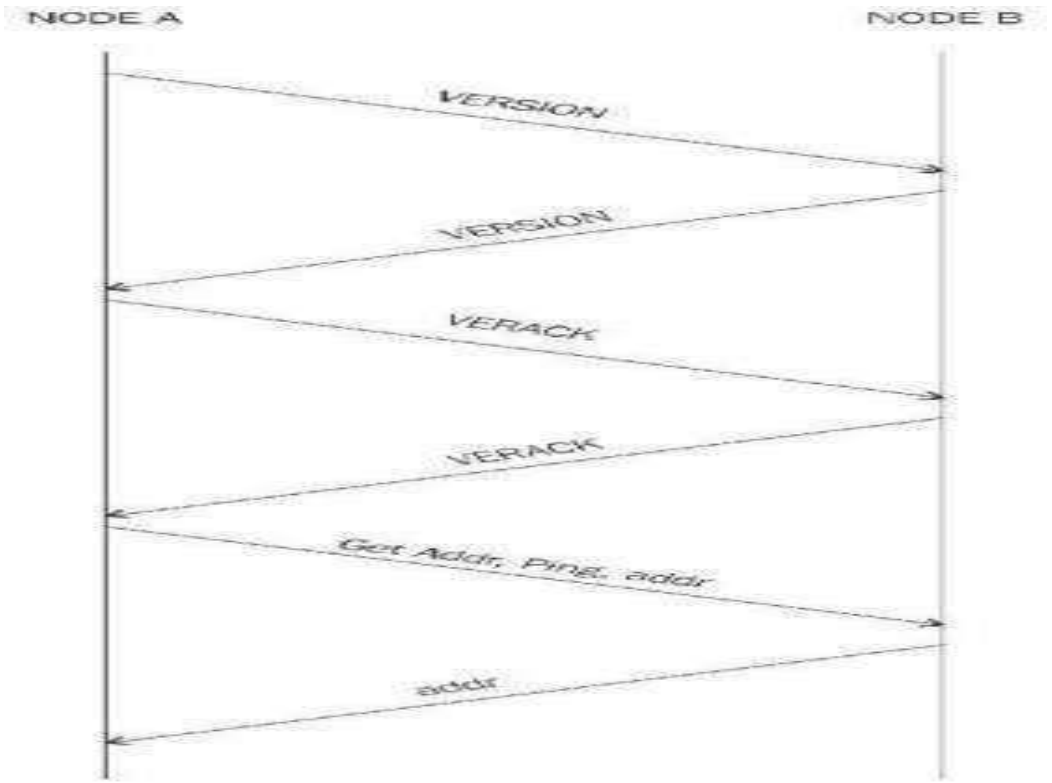
A full node performs four functions: wallet, miner, blockchain, and the network routing node.

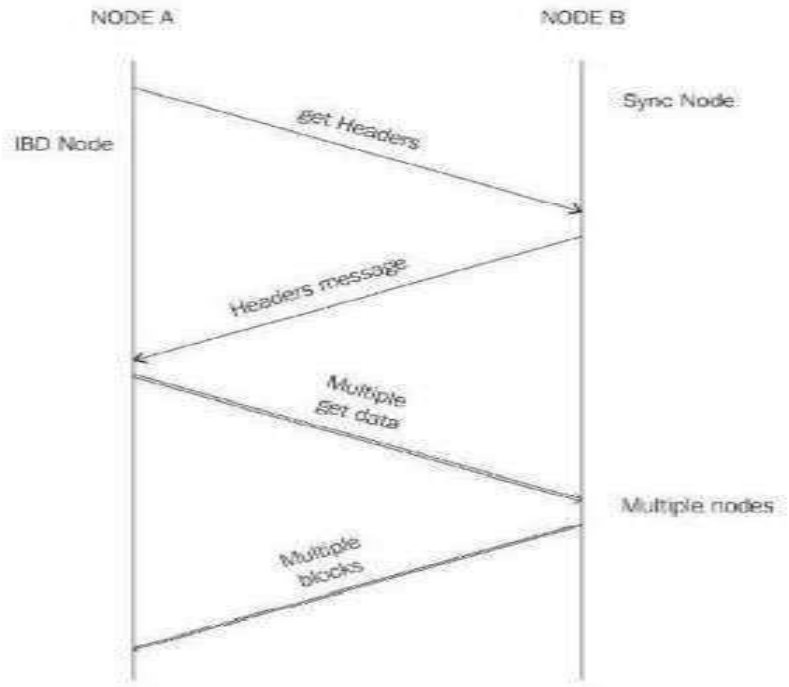
When a bitcoin core node starts up, first, it initiates the discovery of all peers. This is achieved by querying DNS seeds that are hardcoded into the bitcoin core client and are maintained by bitcoin community members. This lookup returns a number of DNS A records. The bitcoin protocol works on TCP port 8333 by default for the main network and TCP 18333 for testnet.

First, the client sends a protocol message Version that contains various fields, such as version, services, timestamp, network address, nonce, and some other fields. The remote node responds with its own version message followed by verack message exchange between both nodes, indicating that the connection has been established.

After this, Getaddr and addr messages are exchanged to find the peers that the client do not know. Meanwhile, either of the nodes can send a ping message to see whether the connection is still live. Now the block download can begin.

If the node already has all blocks fully synchronized, then it listens for new blocks using the Inv protocol message; otherwise, it first checks whether it has a response to inv messages and have inventories already. If yes, then it requests the blocks using the Getdata protocol message; if not, then it requests inventories using the GetBlocks message. This method was used until version 0.9.3





Wallets: The wallet software is used to store private or public keys and bitcoin address. It performs various functions, such as receiving and sending bitcoins. Nowadays, software usually offers both functionalities: bitcoin client and wallet. On the disk, the bitcoin core client wallets are stored as the Berkeley DB file:

:~/.bitcoin$ file wallet.dat

wallet.dat: Berkeley DB (Btree, version 9, native byte-order) Private keys can be generated in different ways and are used by different types of wallets.

Wallets do not store any coins, and there is no concept of wallets storing balance or coins for a user. In fact, in the bitcoin network, coins do not exist; instead, only transaction information is stored on the blockchain (more precisely, UTXO unspent outputs), which are then used to calculate the amount of bitcoins.

WALLET TYPES In bitcoin, there are different types of wallets that can be used to store private keys. As a software program, they also provide some functions to the users to manage and carry out transactions on the bitcoin network.

Non-deterministic wallets :

These wallets contain randomly generated private keys and are also called Just a Bunch of Key wallets. The

bitcoin core client generates some keys when first started and generates keys as and when required. Managing a large number of keys is very difficult and an error-prone process can lead to theft and loss of coins. Moreover, there is a need to create regular backups of the keys and protect them appropriately in order to prevent theft or loss.

Deterministic wallets:

In this type of wallet, keys are derived out of a seed value via hash functions. This seed number is generated randomly and is commonly represented by humanreadable mnemonic code words. Mnemonic code words are defined in BIP39. This phrase can be used to recover all keys and makes private key management comparatively easier.

Hierarchical deterministic wallets :

Defined in BIP32 and BIP44, HD wallets store keys in a tree structure derived from a seed. The seed generates the parent key (master key), which is used to generate child keys and, subsequently, grandchild keys. Key generation in HD wallets does not generate keys directly; instead, it produces some information (private key generation information) that can be used to generate a sequence of private keys. The complete hierarchy of private keys in an HD wallet is easily recoverable if the master private key is known. It is because of this propertythat HD wallets are very easy to maintain and are highly portable.

Brain wallets:

The master private key can also be derived from the hash of passwords that are memorized. The key idea is that this passphrase is used to derive the private key and if used in HD wallets, this can result in a full HD wallet that is derived from a single memorized password. This is known as brain wallet. This method is prone to password guessing and brute force attacks but techniques such as key stretching can be used to slow down the progress made by the attacker. Paper wallets As the name implies, this is a paper-based wallet with the required key material printed on it. It requires physical security to be stored. Paper wallets can be generated online from various service providers, such as https://bitcoinpaperwallet.com/ or https://www.bitaddress.org/.

Hardware wallets:

Another method is to use a tamper-resistant device to store keys. This tamper-resistant device can be custombuilt or with the advent of NFC-enabled phones, this can also be a secure element (SE) in NFC phones. Trezor and Ledger wallets (various types) are the most commonly used bitcoin hardware wallets.

Online wallets :

Online wallets, as the name implies, are stored entirely online and are provided as a service usually via cloud. They provide a web interface to the users to manage their wallets and perform various functions such as making and receiving payments. They are easyto use but imply that the user trust the online wallet service provider.

Mobile wallets :

Mobile wallets, as the name suggests, are installed on mobile devices. They can provide various methods to make payments, most notably the ability to use smart phone cameras to scan QR codes quickly and make payments. Mobile wallets are available for the Android platform and iOS, for example, breadwallet, copay, and Jaxx.

Jaxx Mobile wallet

Bitcoin payments:

Bitcoins can be accepted as payments using various techniques. Bitcoin is not recognized as a legal currency in many jurisdictions, but it is increasingly being accepted as a payment method by many online merchants and e-commerce websites. There are a numbers of ways in which buyers can pay the business that accepts bitcoins. For example, in an online shop, bitcoin merchant solutions can be used, whereas in traditional physical shops, point ofsale terminals and other specialized hardware can be used.

Customers can simply scan the QR barcode with the seller's payment URI in it and pay using their mobile devices. Bitcoin URIs allow users to make payments by simply clicking on links or scanning QR codes. URI (Uniform Resource Idenfier) is basically a string that represents the transaction information. It is defined in BIP21.
The QR code can be displayed near the point of the sale terminal. Nearly all bitcoin wallets support this feature. Business can use the following screenshot to advertise that theycan accept bitcoins as payment.

Various payment solutions, such as xbtterminal and 34 bytes bitcoin POS terminal are available commercially. 34 bytes POS solution.
Bitcoin payment processor, offered by many online service providers, allows integration with e-commerce websites.

Bitcoin investment and buying and selling bitcoins

There are many online exchanges where users can buy and sell bitcoins. This is a big business on the Internet now and it offers bitcoin trading, CFDs, spread betting, margin trading, and various other choices. Traders can buy bitcoins or trade by opening long or short positions to make profit when bitcoin's price goes up or down. Several other features, such as exchanging bitcoins for other virtual currencies, are also possible, and many

online bitcoin exchanges provide this function. Advanced market data, trading strategies, charts, and relevant data to support traders is also available. An example is shown from CEX.IO here.

Bitcoin installation

The bitcoin core client can be installed from https://bitcoin.org/en/download. This is available for different architectures and platforms ranging from x86 windows to ARM Linux, as shown in the following image:

SETTING UP A BITCOIN NODE

A sample run of the bitcoin core installation on Ubuntu is shown here; for other platforms, you can get details from

www.bitcoin.org.

Step 2:

drequinox@drequinox-OP7010:~$ sudo apt-get update

Depending on the client required, users can use either of the following commands, or they can issue both commands at once:

sudo apt-get install bitcoind

sudo apt-get install bitcoin-qt

drequinox@drequinox-OP7010:~$ sudo apt-get install bitcoin-qt bitcoind

Reading package lists... Done Building dependency tree

Reading state information... Done .....

SETTING UP THE SOURCE CODE

The bitcoin source code can be downloaded and compiled if users wish to participate in the bitcoin code or for

learning purpose. Git can be used to download the bitcoin source code:

$ sudo apt-get install git

$ mkdir bcsource

$cd bcsource

drequinox@drequinox-OP7010:~/bcsource $ git clone https://github.com/bitcoin/bitcoin.git

Cloning into 'bitcoin'...

remote:

Counting objects: 78960, done.

remote: Compressing objects: 100% (3/3), done.

remote: Total 78960 (delta 0), reused 0 (delta 0), pack-reused 78957

Receiving objects: 100% (78960/78960), 72.53 MiB | 1.85 MiB/s, done.

Resolving deltas: 100% (57908/57908), done.

Checking connectivity... done.

Change the directory to bitcoin:

drequinox@drequinox-OP7010:~/bcsource$ cd bitcoin

After the preceding steps are completed, the code can be compiled:

drequinox@drequinoxOP7010:~/bcsource/bitcoin$./autogen.sh

drequinox@drequinoxOP7010:~/bcsource/bitcoin$./configure.sh

drequinox@drequinoxOP7010:~/bcsource/bitcoin$make drequinox@drequinoxOP7010:~/bcsource/bitcoin$

sudo make install

SETTING UP BITCOIN.CONF

bitcoin.conf file is a configuration file that is used by the bitcoin core client to save configuration settings. All command line options for the bitcoind client with the exception of -conf switch can be set up in the configuration file, and when bitcoin-qt or bitcoind will start up, it will take the configuration information from that file. In Linux systems, this is usually found in $HOME/.bitcoin/, or it can also specified in the command line using the - conf= switch to bitcoind core client software.

STARTING UP A NODE IN TESTNET

The bitcoin node can be started in the testnet mode if you want to test the bitcoin network and run an experiment. This is a faster network as compared to the live network and has relaxed rules for mining and transactions. Various faucet services are available for the bitcoin test network. One example is Bitcoin TestNet sandbox, where users can request bitcoins to be paid to their testnet bitcoin address. This can be accessed via https://testnet.manu.backend.hamburg/. This is very useful for experimentation with transactions on test net.

The command line to start uptest net is as follows:

bitcoind --testnet -daemon

bitcoin-cli –testnet

bitcoin-qt –testnet

STARTING UP A NODE IN REGTEST

The regtest mode (regression testing mode) can be used to create a local blockchain for testing purposes. The following commands can be used to start up a node in the reg test mode

bitcoind -regtest -daemon

Bitcoin server starting

Blocks can be generated using the following command:

bitcoin-cli -regtest

generate 200 Relevant log messages can be viewed in the .bitcoin/regtest directory on a Linux system under debug.log.

After block generation, the balance can be viewed as follows:

drequinox@drequinoxOP7010:

~/.bitcoin/regtest

$ bitcoin-cli - regtest getbalance 8750.00000000

The node can be stopped using this:

drequinox@drequinox-OP7010:~/.bitcoin$ bitcoin-cli -regtest stop

Bitcoin server stopping

STARTING UP A NODE IN LIVE MAINNET

Bitcoind is the core client software that can be run as a daemon, and it provides the JSON RPC interface. Bitcoin-cli is the command line feature-rich tool to interact with the daemon; the daemon then interacts with the blockchain and performs various functions. Bitcoin-cli calls only JSON-RPC functions and does not perform any actions on its own onthe blockchain.

Bitcoin-qt is the bitcoin core client GUI. When the wallet software starts up first, it verifies the blocks on the disk and then starts up and shows the following GUI:

Bitcoin Core QT client, just after installation, showing that blockchain is not in sync The verification process is not specific to the Bitcoin-qt client; it is performed by the bitcoind client as well. **EXPERIMENTING WITH BITCOIN CLI**

Bitcoin-cli is the command-line interface available with the bitcoin core client and can be used to perform various functions using the RPC interface provided by the bitcoin core client. A sample run of bitcoin-cli getinfo; the same format can be used to invoke other commands A list of all commands can be shown via the following command: Testnet bitcoin-cli, this is just the first few lines of the output, actual output has many commands

HTTP REST: Starting from bitcoin core client 0.10.0, the HTTP REST interface is also available. By default, this runs on the same TCP port 8332 as JSON-RPC.

Bitcoin programming and the command-line interface

Bitcoin programming is a very rich field now. The bitcoin core client exposes various JSON RPC commands that can be used to construct raw transactions and perform other functions via custom scripts or programs. Also, the command line tool Bitcoin-cli is available, which makes use of the JSON-RPC interface and provides a rich toolset to work with Bitcoin.

These APIs are also available via many online service provider in the form of bitcoin APIs, and they provide a simple HTTP REST interface. Bitcoin APIs, such as blockchain.info and bitpay, block.io, and many others, offer a myriad of options to develop bitcoin-based solutions. Various libraries are available for bitcoin programming. A list is shown as follows, and those if you interested can further explore the libraries.

Libbitcoin: Available at https://libbitcoin.dyne.org/ and provides powerful command line utilities and clients.

Pycoin: Available at https://github.com/richardkiss/pycoin, is a library for Python.

Bitcoinj: This library is available at https://bitcoinj.github.io/ and is implemented in Java.

There are many online bitcoin APIs available,the most commonly used APIs are listed as follows:

https://bitcore.io/

https://bitcoinjs.org/

https://blockchain.info/api

All APIs offer more or less the same type of functionality, and it gets difficult to choose which API is the best.

Bitcoin improvement proposals (BIPs) :

These documents are used to propose or inform the bitcoin community about the improvements suggested, the design issues, or information about some aspects of the bitcoin ecosystem. There are three types of bitcoin improvement proposals, abbreviated as BIPs:

Standard BIP: Used to describe the major changes that have a major impact on the bitcoin system, for example, block size changes, network protocol changes, or transaction verification changes.

Process BIP: A major difference between standard and process BIPs is that standard BIPs cover protocol changes, whereas process BIPs usually deal with proposing a change in a process that is outside the core Bitcoin protocol. These are implemented only after a consensus among bitcoin users.

Informational BIP: These are usually used to just advise or record some information about the bitcoin ecosystem, such as design issues.