Mastering Bitcoin

By Andreas M. Antonopoulos

## Preface

#### Writing the Bitcoin Book

I first stumbled upon bitcoin in mid-2011. My immediate reaction was more or less first stumbled upon bitcoin in mid-2011. My immediate reaction was more or less ILO “Pfft! Nerd money!” and ignored it for another six months, failing to grasp its importance. This is a reaction that I have seen repeated among many of the smartest people I know, which gives me some consolation.

The second time I came across bitcoin, in a mailing list discussion, I decided to read the whitepaper written by Satoshi Nakamoto to study the authoritative source and see what it was all about. I still

The realization that “this isn’t money, it’s a decentralized trust network,” started me on a four-month journey to devour every scrap of information about bitcoin I could find. I 1 became obsessed and enthralled, spending 12 or more hours each day glued to a screen, reading, writing, coding, and learning as much as I could.

The topic that had driven me into a frenzy of creativity and consumed mY coin was the topic that had driven me into a frenzy of creativity

#### Intended Audience

Mostly intended for coders.

If You can use a programming language, this book will teach you how cryptographic currencies work, how to use them, and how develop software that works with them.

In-depth introduction to bitcoin for noncoders-those trying to understand the inner workings of bitcoin and cryptocurrencies.

Why Are There Bugs on the Cover?

Highly intelligent and there is no central authority or leader in an ant colony. The highly intelligent апо property from the interaction of the individuals in a social network.

Nature demonstrates that decentralized systems can be resilient and can produce Nature demonstrates that decentralized systems can be resilient aucun proses authority, hierarchy, or complex parts.

#### Code Examples

. All code snippets are available in the GitHub repository (<https://github.com/bitcoinbook/bitcoinbook>) in the code subdirectory of the main repo. Fork the book code, try the code examples, or submit corrections via GitHub.

#### Using Code Examples

#### 

#### Bitcoin Addresses and Transactions in This Book

#### O’Reilly Safari

A membership-based training and reference platform for enterprise, government, educators, and individuals.

Curated playlists from over 250 publishers, incl

#### How to Contact Us

#### Contacting the Author

#### Acknowledgments

It is impossible to make a distinction between the bitcoin technology and the bitcoin community,

Conferences, events, seminars, meetups, pizza gatherings, and small private gather

The first few drafts of the first few chapters were the hardest, because bitcoin is a dif- difficult subject to unravel. Every time I pulled on one thread of the bitcoin technology, I bad to pull on the whole thing. I I repeatedly got stuck and a bit despondent

Struggled to make the topic easy to understand and create a narrative around such a struggled to make the topic

During the development of the book, I made early drafts available on GitHub

#### Early Release Draft (GitHub Contributions)

Make a twitter list with these people!

## Quick Glossary

Contributed definitions have been sourced under a CC-BY license from the bitcoin Wiki (<https://en.bitcoin.it/wiki/Main_Page>)

# Introduction

## What Is Bitcoin?

Unlike traditional currencies, bitcoin are entirely virtual. There are no physical coins or even digital coins per se. The coins are implied in transactions that transfer value from sender to recipient. Users of bitcoin own keys that allow them to prove ownership of bitcoin in the bitcoin network. With these keys they can sign transactions to unlock the value and spend it bY transferring it to a new owner.

Bitcoin in circulation closely follows an easily predictable curve that approaches 21 million by the year 2140. Due to bitcoin’s diminishing rate of issuance, over the long term, the bitcoin currency is deflationary.

#### Digital Currencies Before Bitcoin

Issuers of paper money are constantly battling the counterfeiting problem by using I : increasingly sophisticated papers and printing technology.

Double-spend issues are handled by clearing all | electronic transactions through central authorities that have a global view of the cur- | rency in circulation.

Early digital currency projects issued digital money, usually backed by a { national currency or precious metal such as gold.

They were centralized and, as a | result, were easy to attack by governments and hackers. Early digital currencies used a central clearinghouse to settle all transactions at regular intervals, just like a traditional banking system.

Bitcoin is such a system, decentralized by design, and free of any central authority or point of control that can be attacked or corrupted.

## History of Bitcoin

The Proof-of-Work algorithm (mining) that provides security and resilience for bitcoin has increased in power exponentially, and now exceeds the combined processing T power of the world’s top supercomputers.

The invention itself is groundbreaking and has already spawned new science in the fields of distributed computing, economics, and econometrics.

#### A Solution to a Distributed Computing Problem

A practical and novel solution to a problem in distributed computing, known as the “Byzantine Generals’ Problem.”

Consensus without a central trusted authority represents a breakthrough

To prove the fairness of elections, lotteries, asset registries, digital notarization, and more.

## Bitcoin Uses, Users, and Their Stories

Bitcoin is an innovation in the ancient technology of money. At its core, money simply facilitates the exchange of value between people.

North American low-value retail

North American high-value retail

Offshore contract services

Web store

Charitable donations

Import/export

Mining for bitcoin

## Getting Started

There are many implementations and brands of bitcoin wallets, just like there are brands of web browsers

Also a reference implementation of the bitcoin protocol that includes a wallet, known as the “Satoshi Client” or “Bitcoin Core,” which is derived from the original implementation written by Satoshi Nakamoto.

### Choosing a Bitcoin Wallet

Can categorize bitcoin wallets according to their platform and function and provide some clarity about all the

##### Desktop wallet

-first type of bitcoin wallet crea¯ Running on general-use operating systems

##### Mobile wallet -

most common type of bitcoin wallet. Designed for simplicity and ease-of-use,

##### Web wallet

##### accessed through a web browser and store the user’s wallet on a server owned by a third

Present a compromise by taking control of the bitcoin keys from users in exchange for ease-of-use. It

##### Hardware wallet

operated via USB with a desktop web browser or via near-field-communication (NFC) on a mobile device.

Very secure and suitable for storing large amounts of bitcoin.

##### Paper wallet

even though other materials (wood, metal, etc.) can be used. Paper wallets offer a low-tech but highly secure means of storing bitcoin long term. Offline storage is also often referred to as cold storage.

Another way to categorize bitcoin wallets is by their degree of autonomy and how interact with the bitcoin network:

##### Full-node client

##### A full client, or “full node,” is a client that stores the entire history of bitcoin

Independently validate the entire blockchain and

Consumes substantial computer resources offers complete autonomy (e.g., more than 125 GB of disk, 2 u

##### Lightweight client

##### known as a a simple-payment-verification (SPV) client, connects to bitcoin full nodes (mentioned previously) for access to the bitcoin (mentioned previously) tor

Interact directly with the bitcoin network, without an intermediary.

##### Third-party API client

### Quick Start

Bitcoin addresses start with a 1 or 3. Like email addresses, they can 1 14 be shared with other bitcoin users who can use them to send bitcoin directly to your wallet.

There is nothing sensitive, from a security perspective, about the bitcoin address. Can be posted rity perspective, about the bitcoin address. It can

Without risking the security of the account. Unlike email addresses, you can create new addresses

As you like, all o lets automatically create a new address for every transaction to maximize privacy. A

Address is simply a number that corresponds to a key that she can use to control access to the funds.

### Getting Your First Bitcoin

Tcoin address and any externally identifiable information including transaction posted on the bitcoin ledger, the bitcoin address is simply part of the vast transaction posted on the bitcoin ledger, the bitcoin

Bank account transfers are reversible. For someone selling bitcoin, this difference introduces a very high risk that the buyer will reverse the electronic payment after they have received bitcoin, in effect defrauding the seller.

Methods for getting bitcoin as a new user: new user:

Friend who has bitcoin and buy some from him or her directly. Earn bitcoin by selling a product or service for bitcoin.

Use a bitcoin ATM in your

Use a bitcoin currency exchange linked to your bank account. Ma

### Finding the Current Price of Bitcoin

“Who sets the bitcoin price?” The short answer is that the price is set by mar kets.

A floating exchange rate.

Exchange rate. That mea.. Uctuates according to su demand in the various markets where it is traded. For example, the “price” of bitcoin US dollars is calculated in each market based on the most recent trade of bitcoin

Most bitcoin wallets will automatically convert amounts between bitcoin and other currencies.

### Sending and Receiving Bitcoin

Opens her Mycelium wallet application, and selects Receive. This displays a QR code with Alice’s first bitcoin

A destination bitcoin address

* A destination bitcoin
* Wallet is constantly “listening” to published transactions on the bitcoin network, looking for any that match the addresses in her wallets.

Confirmations:

“Unconfirmed.” This | means that the transaction has been propagated to the network but has not Yet been recorded in the bitcoin transaction ledger, Known as the blockchain. To

In traditional financial terms this is known as clearing. For

# How Bitcoin Works

## Transactions, Blocks, Mining, and the Blockchain

### Bitcoin Overview

Popular blockchain explorers include: blockchain.info

### Buying a Cup of Coffee

Bob’s point-of-sale system will also automatically create a special QR code containing Bob’s point-of-sale system will also automati

Bitcoin network can transact in fractional values, e.g., from millibitcoin (1/1000th of a bitcoin) down to 1/100,000,000th of a millibitcoin (1/1000th of a bitcoin)

Use the term “bitcoin” to refer to any quantity of bitcoin currency,

## Bitcoin Transactions

Has authorized the transfer of that value to another owner. The new owner can now has a transaction tells the network that the owner of some bitcoin value has authorized the transfer of that value to another owner. The new owner can now has authorized the transfer of that value to another owner. The new owner can no another owner, and so on, in a chain of ownership..

### Transaction Inputs and Outputs

Outputs add up to slightly less than inputs and the difference represents an implied transaction fee

“Spending” is signing transaction that transfers value from a previous transaction over to a new owner identified by a bitcoin address.

### Transaction Chains

Uses a previous transactions output as its input. In

Receive change. The transactions form a chain, where the puts from the latest transaction correspond to outputs from previous transactions.

Encumbering” that output with the requirement that Bob produces a signature in order to spend that amount. Th

### Making Change

Outputs that reference both an address of the new owner and an address of the current owner, called the charge address.

(less any applicable transaction fee). Importantly, the change address does not have to be the same address as that of the input and for privacy reasons is often a new address from the owner’s wallet.

They might aggregate many small inputs, or use one that is equal to or larger than the desired payment. Unless the wallet can aggregate inputs in such a way to exactly match the desired payment plus transaction fees,

### Common Transaction Forms

One input and two outputs

Several inputs into a sinoutput (s

Clean up lots of smaller amounts received as change

### Constructing a Transaction

Ortantly, a wallet application can construct transactions even in an envelope, the transaction does not need to be constructed and signed while connected to the bitcoin network

### Getting the right inputs

Because a full-node client takes up a lot of disk space, most user wallets run “lightweight” clients that track only the user’s own unspent outputs.

The wallet application does not maintain a copy of unspent transaction outputs, it can query the bitcoin network to retrieve this information using a variety of APIs available by different providers or by asking a full-node using an application programming interface (API) call. Gramming interface (API) call. Example 2-2

Gramming interface (Ab 1) call. Example

The simple command-line HTTP client cURL to retrieve the response.

$ curl <https://blockchain.info/unspent?active=1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK>

The response in t… conshows one unspent output (one that has not been redeemed yet)

### Creating the Outputs

Transaction output is created in the form of a script that creates an encumbrance on the value and can only be redeemed by the introduction of a solution to the script.

This (fee) is not explicit in the transaction; it is implied by the difference between inputs and outputs.

### Adding the Transaction to the Ledger

#### Transmitting the transaction

With each bitcoin client participating by connecting to several peer-to-Peer network, with each bitcoin client participating by connecting to several other bitcoin clients. The purpose of the bitcoin network is to propagate transactions and blocks to all participants.

#### How it propagates

Bitcoin wallet does not have to be connected to Bob’s bitcoin wallet directly and she does not have to use the

Any bitcoin node that receives a valid transaction it has not seen before will immediately forward it to all other nodes to which it is connected, a propagation technique known as flooding. Thus, the transaction rapidly propagates out across the peer-to peer network, reaching a large percentage of the nodes within a few seconds.

#### Bob’s view

#### wallet application can also independently verify that the transaction is well formed, uses previously unspent inputs, and contains sufficient transaction fees to be included in the next block

For a new block, or up to 60 mSIX confirmations. Although confirmations ensure the transaction has been accepted bY the whole network, such a delay is unnecessary for small-value items such as a cup of cup of coffee. A merchant

## Bitcoin Mining

Mining nodes validate all transactions by reference to bitcoin’s consensus rules. Mining nodes validate all transactions by reference to ditcoms consensus or malformed transactions.

Amount of bitcoin created per block is limited and diminishes with time, following a fixed issuance schedule.

Between cost and reward. Mining uses electricity to new bitcoin and transaction fees. However, the reward will only be collected if the miner has correctly validated all the transactions, to the satisfaction of the rules of Consensus.

Finding such a solution, the sorace to find a solution to a block of transactions. Finaing SUCI a SolutiŪTI, UITU JU across the entire bitcoin network. The algorithm for Proof-of-Work involves repeatedly hashing the header of the block and a random number with the SHA256 cryptographic algorithm until a solution matching a predetermined pattern emerges. The first miner to find such a solution wins the round of competition and publishes that block into the blockchain.

Essentially hundreds of mining algorithms printed, running in parallel on a single silicon chip.

## Mining Transactions in Blocks

As these are seen by the bitcoin network nodes, they get added to a temporary pool of unverified transactions maintained by each node. A

Prioritized by the highest-fee transactions Transactions are added to the new block, prioritized by the nignest-fee transactions transactions as soon as he receives the previous block from the network, knowing he has lost that previous round of competition. He

Was picked up dy the work and included in the pool of unveri- 1 block, called a candidate block, generated by Jing’s mining pool.

Lock « Jing’s winning block became part of the blockchain as block #277316, containing 419 transactions, including Alice’s transaction. The block containing Alice’s transaction is counted as one “confirmation” of that transaction.

Further assurance, as they pile on more computation in a is considered irrevocable, because it would require an immense amount of computation to invalidate and recalculate six blocks.

## Spending the Transaction

# Bitcoin Core: The Reference Implementation

That first implementation, then simply known as “Bitcoin” or “Satoshi client,” has been heavily modified and improved. It has evolved into what is known as Bitcoin Core, to differentiate it from other compatible implementations. Bitcoin Core is the reference implementation of the bitcoin system, implementations. Bitcoin Core is the reference implementation of the bitcoin system, should be implemented.

Wallets, a transaction and block validation engine, and a full network node in the peer-to-peer bitcoin network.

## Bitcoin Development Environment

## Compiling Bitcoin Core from the Source Code

Denoted by a $ symbol. In the examples, when you see text after a $ denoted by a $ symbol. In the examples, when you see text after a $ immediately following it, then press Enter to execute the command.

$ git clone <https://github.com/bitcoin/bitcoin.git>

Change to this directory by typing source code repository in $ cd bitcoin

### Selecting a Bitcoin Core Release

By default, the local copy will be synchronized with the most recent code, which might be an unstable or beta version of bitcoin. Before compiling the code, select a specific version by checking out a release tag. Used by the developers to mark special use cases UID

Release candidates, which are intended for testing, have the suffix “rc” Stable releases that can be run on production systems have no suffix. To synchronize the local with this version, use the git checkout command: $ git checkout v0.15.0

You can confirm you have the desired version “checked out” by issuing the command rou can comm

### 

### Configuring the Bitcoin Core Build

Build the command-line bitcoin client, also known as page. In this chapter, we will build the command-line Bitcoin client, also known as bitcoind on Linux. Review the instructions for compiling the bitcoind command-line client on your platform by typing more doc/build-unix.md Alternative instructions for macOS and Windows can be found in the doc directory, as build-osx.md or build-windows.md, respectively.

Carefully review the build prerequisites, which are in the first part of the build documentation

Assuming the prerequisites installed, you start the build process by generating a set of build scripts using the autogen.sh script.

Creates a set of automatic configuration scripts that will interrogate your system to discover the correct settings and ensure you have all the necessary libraries to compile the code.

All the default features. We won’t be using the configuration flags, but you should review them to understand what optional features

Run the configure script to automatically discover all the necessary libraries

### Building the Bitcoin Core Executables

Next, You will compile the source code, a process that can take up to an hour to complete, depending on the speed of your CPU and available memory. Dur

Type make to start compiling

Step is to install the various executables on your system using the make install command

## Running a Bitcoin Core Node

Bitcoin Core keeps a full copy of the blockchain by default, with every transaction that has ever occurred on the bitcoin network since its inception in 2009

On the speed of your CPU and internet connection. Bitcoin Core will not be able to process transactions or update |

Pace, bandwidth, and time to complete the initial synchronization. You can configure Bitcoin to reduce the size of the blockchain by discarding old blocks before discarding data.

Why would you want to run a node? Here are some of the most common reasons:

(API) access to the network and blockchain.

### Configuring the Bitcoin Core Node

Will look for a configuration file in its data directory on every start. In

To locate the configuration file, run bitcoind -printtoconsole in your terminal and look for the first couple of lines.

More than 100 configuration options that modify the behavior of the network node, the storage of the blockchain, and many other aspects of its operation. To see a listing of these options, run bitcoind –help:

Transaction Database Index and txindex Option Transaction Database Index and txindex Option

By default, Bitcoin Core builds a database containing only the transactions related to 1 By default, Bitcoin Core bullas a database containing only the transactions related 10 getrawtransaction (see “Exploring and Decoding Transactions” on page 45), you achieved with the txindex option. Set txindex=1 in the Bitcoin Core configuration

Ctrl-C to interrupt the process

To run Bitcoin Core in the background as a process, start it with the daemon option, as bitcoind-daemon.

To monitor the progress and runtime status of your bitcoin node, use the command bitcoin-cli getblockchaininfo:

Node currently fetches the block headers of the best chain and afterward continues to download the full blocks.

You should add bitcoin to the startup scripts in your operating system, so that it runs continuously bitcoin to the startup scripts in your operating

## Bitcoin Core Application Programming Interface (API)

JSON-RPC interface that can also be accessed using the command-line helper bitcoin-cli.

Detailed description, and information on the parameters, add the command name after help

### Getting Information on the Bitcoin Core

The most important commands include getblockchaininfo, getmempoolinfo, getnetworkinfo and getwalletinfo

### Exploring and Decoding Transactions

Commands: getrawtransaction, decoderawtransaction

E API to retrieve and examine that transaction by passing the transaction ID as a parameter:

Is not authoritative until a transaction has been A transac

Not mean the transaction was not pressed. This is known as fied prior to confirmation in a block. After confirmation, the txid is immutable and authoritative.

Getrawtransaction returns a serialized transaction in hexadecimal

“Vin” : “vout”: [

### Exploring Blocks

Commands: getblock, getblockhash

Blocks can be referenced either by the block height or by the block hash.

### Using Bitcoin Core’s Programmatic Interface

Access functions programmatically.

RPC stands for Remote Procedure Call, which means that we are calling procedures (functions) that are remote (on the Bitcoin Core node) via a network protocol,

Bitcoin Core will create a random password on each start and place it in the data’ directory under the name .cookie. The bitcoin-cli helper can read this password

And pass it to curl password with the helper script provided in ./share/rpcuser/rpcuser.py in

There are libraries in most every programming language that “wrap” the Core API in a way that makes this a lot simpler.

## Alternative Clients, Libraries, and Toolkits

### C/C++

### JavaScri

### Java

### PHP

### Python

### Ruby

### Go Rust

### Objective-C

## Keys, Addresses

Cryptography can also be used to prove knowledge of a secret without revealing that secret (digital signature), or prove the authenticity of data (digital fingerprint).

transaction data are not encrypted and do not need to be encrypted to protect the funds.

### Introduction

Ownership of bitcoin is established through digital kevs, bitcoin addresses, and digital signatures. The digital keys are not actually stored in the network, but are instead created and stored by users in a file, or simple database, called a wallet. The

Can be generated and managed by the user’s wallet software without reference to the blockchain access to the internet.

The digital signature used to spend funds is also referred to as a witness, a term used in cryptography. The witness data in a bitcoin transaction testifies to the true ownership of the funds being spent.

Cases, a bitcoin address is generated from and corresponds to a prom my as scripts

### Public Key Cryptography and Cryptocurrency

Prime number exponentiation and elliptic curve multiplication, have tions, such as These mathematical functions are practically irreversible, meaning that they are easy to calculate in one direction and infeasible to calculate in the oppo that they are

Bitcoin uses elliptic curve multiplication as the basis for its cryptography.

We use public key cryptography to create a key pair that controls a 1 1 key

Spending bitcoin, the current bitcoin owner presents her public key and a signature (different each time, but created from the same private key) in a transaction to spend those bitcoin. Through the presentation of the public key and signature, everyone in the bitcoin network can verify and accept the transaction as valid, confirming the person transferring the bitcoin owned them at the time of the transfer.

Stored together as a key pair for convenience. However, the public key can be calculated from the private key, so storing only the private key is also possible.

### Private and Public Keys:

A one-way cryptographic function, to

### Private Keys

A private key is simply a number, picked at random. Owner

Used to create signatures that are required to spend bitcoin by proving ownership of funds used in a transaction.

0 giving backed up and protected from accidental loss, because if it’s lost it cannot be recovered and the funds secured by it are forever lost, too.

Can pick your private bitcoin private key is just a number. You can pick your private times and you have the binary digits of a random private key

#### Generating a private key from a random number

Most important step in generating keys is to find a secure source of entropy; or randomness.

Creating a bitcoin key is essentially the same as “Pick a num entropy, or randomness. Creating a bitcoin key is essentially the same as rick a number as long as it is not predictable or repeatable.

Do not write Your own code to create a random number or use a simple” random number generator offered by your programming language. Use a cryptographically secure pseudorandom number, generator (CSPRNG) with a seed from a source of sufficient entropy.

Bitcoin’s private key space, (2256) is an unfathomably large number. It is approximately 107 in decimal. For comparison, visible universe is estimated to contain 1080 atoms.

The dumpprivkey command does not generate a private key from a public key, as this is impossible. The command simply reveals the private key that is already known to the wallet and which was generated by the getnewaddress command.

### Public Keys

The generator point, and K is the resulting public key. The reverse operation, known as “finding the discrete logarithm”-calculating k if you know K-is as difficult as trying all possible values of k, logarithm-calculating k

Curve multiplication is a type of function that cryptography (multiplication) and impossible to do in the reverse direction division

### Elliptic Curve Cryptography Explained

Figure 4-2. An elliptic curve

Bitcoin uses a specific elliptic curve and set of mathematical constants, as defined in a standard called secp256k1, established by the National Institute of Standards

Mod p (modulo prime number p) indicates that this curve is over a finite field of prime order p&gt; also written as ¥ p&gt; where p = 225 - 232 - 29 - 28 - 27 - 26 − 2ª − 1

Because this curve is defined aver a finite field of prime order instead of over the real numbers, it looks like a pattern of dots scattered in two dimensions, which makes it difficult to visualize.

There is a point called the “point at infinity,” which roughly c corresponds to the role of zero in addition.

* Is associative, which means that (A + B) + C = A + (B+ C). That means we can write A + B + C without parentheses and without ambiguity.
* We can define multiplication in the standard way that extends addition

### Generating a Public Key

A private key in the form of a randomly generated number k, we multianother point somewhere else on the curve, which is the corresponding public key K.

Always the 1 The generator point is specified

On the curve. Because the generator point is always the same for all bitcom

A bitcoin address (derived from K) can be shared with anyone and does not reveal the user’s private key (k).

Public key K is defined as a point K = (X,Y): (%,¥):

Our goal is to find the multiple kG of the generator point G, which is the same as adding G to itself, k times a row. In elliptic curves, adding a point to itself is the equivalent of drawing a tangent line on the point and finding where it intersects the curve again, then reflecting that point on the x-axis.

Implementations use the OpenSSL cryptographic library (<http://bit.ly/1q17bn8>) to do the elliptic curve math. For

Figure 4-4. Elliptic curve cryptography: visualizing the multiplication of a point G by an integer k on an elliptic curve

## Bitcoin Addresses

Addresses produced from public keys consist of a string of numbers and letters, beginning with the digit “1.”

Function that produces a fingerprint or “hash” of an arbitrary-sized input. Cryptographic hash functions are used extensively in bitcoin: in bitcoin addresses, in script addresses, and in the mining Proof-of-Work algorithm. The algorithms used to make a bitcoin address from a public key are the Secure Hash Algorithm (SHA) and the RACE Integrity Primitives Evaluation Message Digest (RIPEMD), specifically SHA256 and RIPEMD160.

A bitcoin address is not the same as a public key. Bitcoin addresses are derived from a public key using a one-way function.

Addresses are almost always encoded as “Base58Check” which uses 58 characters (a Base58 number sysBase 58Check Encoding” on page 66), which uses 58 characters (a Daseo mum sys } = ง errors in address transcription and entry. Base58Check is also used in many other ways in bitcoin, whenever there is a need for a user to read and correctly transcribe a

### Base58 and Base58Check Encoding

In order to represent long numbers in a compact way, using fewer symbols, many order to represent long numbers in a compact way, using fewer symbols, many I computer systems It offers a balance between compact representation, readability, and detection and prevention.

Base58 is a subset of Base64, using upper- and lower case letters and numbers, but omitting some characters that are frequently mistaken for one another and can appear identical when displayed in certain fonts. Without the O (number zero), O (capital 0), 1 (lower L), I (capital i), cr and the symbols “+” and “/”. Or, more simply, it is a set of lowercase and capital letters the symbols “+” and “/”.

Has a built-in error-checking encoding format, frequently used in bitcoin, which has a Dun-in error-checking ɔeing encoded. The checksum is derived from the hash of the encoded data

This prevents mistyped bitcoin address from being accepted by the wallet software as a valid i destination, an error that would otherwise result in loss of funds.

We first add a prefix to the data, called the “version byte,” which serves to easily identify the type of data that is encoded. For example, in the case of a bitcoin address the prefix is zero (0x00 in hex), encoded. For example, in the case of a bitcoin address the prefix is zero (0x00 in hex), whereas the prefix used when encoding a private key is 128 (0x80 in hex). A list of

Compute the “double-SHA” checksum, meaning apply the SHA256 hashalgorithm twice on the previous result (prefix and data): checksum = SHA256(SHA256(prefix+data))

Resulting 32-byte hash (hash-of-a-hash), we take only the first four bytes. These four bytes serve as the error-checking code, or checksum.

To read, and easy to detect errors. The version pa III Dascoe ase58 contain specific characters at the beginning of the Base58Check-encoded payoad. These characters make it easy for humans to identify the type of data

Table 4-1. Base58Check version prefix and encoded result examples

### Key Formats

Representations all encode the same number, even though they look different. .

Hexadecimal and raw binary formats are used internally in software and rarely shown to decimal and raw binary formats are used internally in software and rarely shown to users.

The WIF is used for import/export of keys between wallets and often used QR code (barcode) representations of private keys.

Table 4-3 shows the private key generated in these three formats.

Note that the “raw binary” is not shown in Table 4-3 as any encoding. Display here would, by definition, not be raw binary data.

Wif-to-ec command from Bitcoin Explorer (see

#### Decode from Base58Check

iyiväⱭ, ɩne WIF version p\*\*\* Notice that the “payload” of the compressed key is appended

#### Encode from hex to Base58Check

#### Encode from hex (compressed key) to Base58Check

#### Public key formats

Public keys are also presented in different ways, usually as either compressed or uncompressed public keys.

Prefix 04 followed by two 256-bit numbers: one for the x coordinate of the point, the other for the y coordinate.

#### Compressed public keys

Compressed public keys were introduced to bitcoin to reduce the size of transactions

Each public key requires 520 bits (prefix + x + y), which

That allows us to store only the x coordinate of the public key point, omitting the y coordinate and reducing the size of the key and the space required to store it by the y coordi

There are two possible prefixes: because the left side of the equation is %, the solution for y is a square root, which can have a positive or negative value. Visually,

While we can omit the y coordinate we have to store the sign of ¥ (positive or negative); or in other words, we have to remember if it was above or below the x-axis

To distinguish between the two possible values of y, we store a compressed public key with the prefix 02 if the y is even, and 03 values of y, we store a compressed public key with une prema voi une y 10 womens coordinate and uncompress the public key to the full coordinates of the point. Public

If we convert this compressed public key to a bitcoin public key to a bitcoin address using the double-hash function (RIPEMD160 (SHA256(K))) it will produce a different bitcoin address.

Not all clients support compressed public keys yet.

#### Compressed private keys

Formats are not used interchangeably. In a

Goal here is to signal to the wallet importing these private keys whether it must search the blockchain for compressed or uncompressed public keys addresses

If a bitcoin wallet is able to implement compressed public keys, it will use those in all transactions.

## Implementing Keys and Addresses in C++

Example 4-3 shows the complete step-by-step process, from private key to Base58Check-encoded bitcoin address.

## Implementing Keys and Addresses in Python

Most comprehensive bitcoin library in Python is pybitcointools (https :// [github.com/vbuterin/pybitcointools](http://github.com/vbuterin/pybitcointools)) by Vitalik Buterin. In

Using the Python ECDSA library for the elliptic curve math and without using any specialized bitcoin libraries.

## Advanced Keys and Addresses

### Encrypted Private Keys (BIP-38)

The need for confidentiality of the private keys is a truism that is quite difficult to achieve in practice, because it conflicts with the equally truism that is quite difficult to achieve in practi

A USB nasn univer the backup itself is stolen or lost? Thes flicting security goals led to the introduction of a portable and convenient stanu and bitcoin clients, standardized by BIP-38 (see Appendi:

Cheme is a Base58Check-encoded encrypted private key that veginis VY JUA “ prefix 6P. If you see a key that starts with 6P, it is encrypted and requires passphrase in order to convert (decrypt)

### Pay-to-Script Hash (P25H) and Multisig Addresses

Designate the beneficiary of a bitcoin transaction as the hash of a script, instead of the owner of a public key.

Requirements are designated time the address is created, within the script, and all inputs to this address will be encumbered with the same requirements.

P2SH is not necessarily the same as a multisignature standard transaction. A P2SH address most often represents a multinature script, but it might also represent a script encoding other } types of transactions.

### Vanity Addresses

M signatures (also known as the “threshold”) from a total N keys, known as an M-of-N multisig, where M is equal to or less than N. Picking a private k at random, deriving the public key, deriving the bitcoin address, and checking to see it matches the desired vanity pattern rpe=ating billions of times until a match is found.

No less or more secure than any other address.

Example 4-9. Vanity address miner

users are likely to look at the vanity pattern and a few characters beyond, for example noticing the “1Kids33” part of the

Users are likely to look at the vanity pattern and a few characters beyond, for example noticing the “1Kids33” part of the

What is affordable for Eugenia becomes unaffordable for the attacker, especially if the potential reward of fraud is not high enough to cover the cost of the especially if the potential reward Personal computer and expensive even with a custom vanity pool.

### Paper Wallets

Bitcoin private keys printed on paper.

If the paper wallet keys are generated offline and never stored on a computer system, they are much more secure against hackers, keyloggers, and other online computer threats.

Generator at bitaddress.org. This page contains all the code necessary to generate keys and paper wallets, even while completely disconnected from the internet. To use it, save the HTML page on your local drive or on an external USB flash drive. Disconnect from the internet and open the file in a browser. The disadvantage of a simple paper wallet system is that the printed keys are vulnera ble to theft.

Although you can deposit funds into a paper wallet several times, Although you can deposit funds into a paper wallet several times,

Reduce the risk of key compromise.

# Wallets

## Wallet Technology Overview

A common misconception about bitcoin is that bitcoin wallets contain bitcoin. In fact, the wallet contains only keys. The “coins” are recorded in the blockchain on the bitcoin network.

In the form of transaction outputs (often noted as vout blockcŋau Л

First type is a nondeterministic wallet, where each key is independently generated The first type is a nondeterministic wallet, where each key is independently generateu also known as a JBOK wallet from the phrase “Just a Bunch Of Keys.

Second type of wallet is a deterministic wallet, where all the keys are derived from The second type of wallet is a deterministic wallet, where all the keys are derived from a single master key, known as the seed. All the keys in this type

Uses a tree-like structure and is known as a monly used derivation methou ลง น

Initialized from a seed. To make these easier to use, seeds are encoded as English words, also known as mnemonic code words.

## Nondeterministic (Random) Wallets

The disadvantage of random keys is that if you generate many of them You must keep copies of all of them. Meaning that the wallet must be backed up frequently. Each key must be

Address reuse reduces privacy bY associating multiple transactions and addresses with each other.

### Deterministic (Seeded) Wallets

Derived from a common seed, through the use of a one-way hash function. The seed is a randomly generated number that is combined with other data, such as an index Is a randomly generated 1 I

A single backup at creation time is sufficient. The seed is also sufficient for a wallet export or import, allowing for easy migration of all the user’s keys between different wallet implementations.

### HD Wallets (BIP-32/BIP-44)

Wallets offer two major advantages over random (nondeterministic) keys. First, the tree structure can be used to express additional organ when a specific branch of subkeys is used to receive incoming payments and ent branch is used to receive change from outgoing payments.

Specific functions, or accounting categories.

Users can create a sequence of public keys without having access to the corresponding private keys.

### Seeds and Mnemonic Codes (BIP-39)

### Wallet Best Practices

Industry standards have emerged that make bitcoin wallets broadly interoperable, easy to use, secure, and flexible.

. A user can export a mnemonic generated on one of these wallets and import it in another wallet, recovering all transactions, keys, and addresses.

### Using a Bitcoin Wallet

The sequence of words is important, so

A 12-word mnemonic is shown in Table 5-1, for simplicity. In fact, most hardware wallets generate a more secure 24-word mnemonic.

## Wallet Technology Details

Mnemonic Code Words (BIP-39)

Mnemonic words are often confused with “brainwallets.” They are Mnemonic words are often confused with “brainwallets.” They are not the same. The primary difference is mat a brainwallet consists of words chosen by the user

Makes mnemonic words much more secure, because. Humans are very poor sources of randomness.

1. Create a random sequence (entropy) of 128 to 256 bits.
2. Create a checksum o the random sequence by taking the first (entropy-1
3. Add the checksum to the end of the random sequence

4. Split the result into 11-bit length segments.

5. Map each 11-bit value to a word from the predefined dictionary of 2048 words.

6. The mnemonic code is the sequence of words.

Figure 5-6. Generating entropy and encoding as mnemonic words

Purof a salt in a key-stretching function is to make it difficult to build a lookup table of a salt in a key-stretching function is to make it difficult to build a 100^up LUMIL allows the introduction of a passphrase that serves as an additional security factor

PBKDF2 stretches the mnemonic and salt parameters using 2048 rounds of hashing with the HMAC-SHA512 algorithm, producing a 512-bit value as its final output. That 512-bit value is the seed.

Makes it extremely costly (in com-t putation) to try more than a few thousand passphrase and mnemonic combinations, while the number of possible derived seeds is vast (2512).

The optional passphrase creates two important features:

Nakes a mnemonic useless on its A second factor (something memorized) that makes a memo

A form of plausible deniability or “duress wallet,” where a chosen passphrase leads to a wallet with a small amount of funds used to distract an attacker from the “real” wallet that contains the majority of funds.

The use of a passphrase also introduces the risk of loss:

If the owner backs up the passphrase in the same place as the seed, it defeats the purpose of a second factor.

### Creating an HD Wallet from the Seed

Is deterministically derived from this root seed, which makes it possible to re-create the entire HD wallet from that seed in any compatible makes it possible to re-create the entire HD wallet from that seed in any compa taining thousands or even millions of keys

HMAC-SHA512 algorithm and the resulting hash is used to create a master private key (m) and a master chain code ©.

Introduce deterministic random data to the process, so that chain code is used to introduce deterministic random data to the process,

Figure 5-10. Extending a parent private key to create a child private key

Repeating the process one level down the tree, each child can in turn become a parent and create its own children, in an infinite number of generations.

Child private keys are indistinguishable from nondeterministic (random) keys.

You need both the child private key and the child chain code to start a new branch and derive grandchildren.

The key and chain code, and combined these are called an extended key. The term “extended key” could also be thought of as “extensible key” because such a key can be used to derive children.

Very useful characteristic of HD wallets is the ability to I derive public child keys from public parent keys, without having the private keys, This gives us two ways to derive a child public key: either from the child private key, or directly from the parent public key.

Can produce an infinite number of public keys and bitcoin addresses, but cannot spend any of the money sent to those addresses. M

Common application of this solution is for cold-storage or hardware wallets.

The user can create “receive” addresses at will, while the private keys are safely offline. To spend the funds, the user can use the extended private key on an offline signing bitcoin client or sign transactions on the hardware wallet device

### Using an Extended Public Key on a Web Store

14 attracted many orders from address, it became difficult to correctly match orders and transactions, espeially when multiple orders for the same amount came in close together.

Hardware wallets will never export private keys-those always remain on the device.

Mycelium Gear, which is an open source web-store plugin for a variety of web hosting and con

Hardened child key derivation

Leaked, it can be used with the chain code to derive all the other chúa priall the private keys of all the children. Worse, the child private key together with a parent chain code can be used to deduce the parent private key.

An alternative derivation function called hardened derivation, which “breaks” the relationship between parent public key and child ened derivation, which “breaks” the relationship between parent public key ana cnia chain code. The hardened derivation function uses the parent private key to uenive parent/child sequence,

Function looks almost identical to the normal child private key derivation, except that the parent

Completely different from

Create a “gap” in the tree above the level where extended public keys are used.

Level-1 children of the master keys always derived through the hardened deriva

Index numbers for normal and hardened derivation

Index numbers between 0 and 2-1 (0x0 to 0x7FFFFFFF) are used only for normal derivation. Index numbers between 231 and 232-1 (0x80000000 to 0xFFFFFFFF) are used only for hardened derivation.

The first hardened child (index 0x80000000) is displayed a as O’. In sequence then, the second hardened key would have index Ox80000001 and would be displayed as 1’, and so on. When you see an wallet index i’, that means 231+i.

HD wallet key identifier (path)

A “path” naming convention, with each level of the tree separated by a slash (/) character (see Table 5-6). Private keys derived from Master private key start with “m.” Public keys derived from the master public key start with “M.” Therefore, the first child private key of the master private key is m/0- The first child public key is M/O. The second grandchild of the first child is m/0/1,

Navigating the HD wallet tree structure dened children. As deep as you want, with an infinite number of generations. W

Index as a special identifier that signifies the “purpose” of the tree structure.

BIP-44 proposes a multiaccount structure as “purpose” extending that specification,

Only used one branch of the tree: m/44’/. “Purpose” is always set to 44’.

Five predefined tree levels: :

M / purpose’ / roin tune’ / account’ / change / address\_index

the fourth level, “change,” an HD wallet has two subtrees, one for creating receiving addresses and one for creating change addresses.

# Transactions

## Introduction

Transactions are data structures that encode the transfer of value between participants in the bitcoin system.

The term “wallet” in this chapter, we are referring to the software that constructs transactions, not just the database of keys.

## Transactions in Detail

### Transactions-Behind the Scenes

In bitcoin, there are no coins, no senders, no recipients, no bola no accounts, and no addresses. All those things are constructed at a higher level for the benefit of the user, to make things easier to understand.

### Transaction Outputs and Inputs

Transaction outputs are indivisible chunks of bitcoin currency, recorded on the blockchain, and recognized as valid by the entire network. Bitcoin full nodes track all blockchain, and recognized as valid by the entire network. Bitcoin full nodes track all available and spendable outputs, known as unspent transaction outputs, or UTXO.

All UTXO is known as the UTXO set and cu

Grows as new UTXO is created and shrinks when UTXO is consumed. Every transaction represents a change (state transition) in

Any arbitrary value, once created it is indivisible. This

Outputs are discrete and indivisible units of= of outputs that needs to be empnasi-zeu

Only be consumed in its entirety by a transaction.

Chunks of bitcoin value move forward from owner to owner in a chain of transac

The exception to the output and input chain is a special type of transaction called the coinbase transaction, which is the first transaction in each block

### Transaction Outputs

Consist of two parts:

Amount of bitcoin,and a cryptographic puzzle is also known as a docking script, a witness script, or a scriptPubKey.

In the JSON encoding, the outputs are in an array (list) named vout:

#### Transaction serialization-outputs

#### Transmitted over the network or exchanged between applications sentation of a data structure into a format that can be transmitted one byte at a time, also known as a byte stream.

#### 

#### Bitcoin libraries store transactions internally in data structures (usually object-oriented structures).

#### Deserialization or transaction library’s have built-in functions

#### Encoded in little-endian In the serialized transaction

### Transaction Inputs

The first part of an input is a pointer to an UTXO by reference to the transaction hash and an output index, which identifies the specific UTXO in that transaction. The second part is an unlocking script, which the wallet constructs in order to satisfy the spending conditions set ing script, which

Not all unlocking scripts contain signatures. The third part is a sequence number, which will be discussed later.

An array (list) called vin:

List (because one UTXO

* Transaction ID,
* An output index (vout)
* A scriptsig,
* A sequence number

Not just Alice’s wallet that needs to retrieve UTXO referenced in the inputs. Once this transaction is broadcast to the network, every validating node will need to retrieve the UTXO referenced in the transaction inputs in order to validate the transaction

To calculate the amount paid in fees, you must the sum of the values of inputs and outputs. But without retrieving the UTXO erenced in the inputs, you do not know their value. So a seemingly simple opera like counting fees in a single transaction in fact involves multiple steps and data from multiple transactions.

Transaction ID is serialized in reversed byte order,

Output index is a 4-byte group of zeros easy to identify

Length of the scriptSig is 139 bytes, or 8b in hex

Sequence number is set to FFFFFFFF, again easy to identify

### Adding Fees to Transactions

Fees also serve as a security mechanism themselves, bY making it economically infeasible for attackers to flood the network with transactions.

Fees are calculated based on the size of the transaction in kilobytes, not the value of the transaction in bitcoin.

Transaction fees affect the processing priority, meaning that a transaction with sufficient fees is likely to be included in the next block mined, whereas a transaction with insufficient or no fees might be delayed, processed on a best-effort basis after a few blocks, or not processed at all. Transaction fees are not best-effort transactions without fees might be processed eventually; however, including transaction fees encourages priority processing.

Capacity limits in bitcoin have created competition between transactions, resulting in higher fees and effectively making free transactions a thing of the past. Zero fee or very low fee transactions rarely get mined and sometimes will not even be propagated across the network.

Fee relay policies are set by the minrelaytxfee option. The current default minrelaytxfee is 0.00001 bitcoin or a hundredth of a millibitcoin per kilo

Dynamic fees can be implemented through a third-party fee estimation service or with a built-in fee estimation algorithm. If

(average or median fee in the last block) to sophisticated (statistical analysis). They (average or median fee in the last block) to sophisticated (statistical analysis) probability of being selected and included within a certain number of blocks.

## Adding Fees to Transactions

\data structure of transactions does pot have a field for fees. Instead, fees are implied as the difference between the sum of inputs and the sum of outputs.

Fees = Sum(Inputs) - Sum(Outputs)

A transaction with that many inputs will be larger than one kilobyte, perhaps

Median-sized transaction.

The transaction and multiplying that by the per-kilobyte fee. Many wallets will overpay fees for larger transactions to ensure the transaction is processed promptly.

## Transaction Scripts and Script Language

The bitcoin transaction script language, called Script, is a Forth-like reverse-polish notation stack-based execution language.

Both the locking script placed on an UTXO and the unlocking script are

When a transaction is validated, the unlocking written in this scripting language. When a transaction is validated, the unlocking written in this scripting language. When a transaction 15 satisfies the spending condition.

Designed to be limited in scope and executable on a range of hardware, perhaps as simple as an embedded device.

Requires minimal processing and cannot do many of the fancy things modern programming languages can do. For its use in validating programmable money, this is a deliberate security feature.

Locking scripts can be written to express a vast variety of complex conditions.

Validation is not based on a static pattern, but instead is achieved through the execution of a scripting language.

### Turing Incompleteness

-there are no loops or complex flow control capabilities other than conditional flow control. This ensures that the language is not Turing Complete, meaning that scripts have limited complexity and predictable execution times.

Cannot be used to create an infinite loop or other form of “logic bomb” that could be embedded in a transaction in a way that causes a denial-of-service attack against the bitcoin network.

### Stateless Verification

Execution of the script, or state saved after execution of the script.

Information needed to execute a script is contained within the script. A script will &gt;redictably execute the same way on any system.

Every other system in the bitcoin network wiŋ also verity the sumpis predictability of outcomes is an essential benefit of the bitcoin system.

### Script Construction (Lock + Unlock)

The source code refers to the unlocking script as scriptSig. You will also see the unlocking script referred to as a witness (see “Segregated Witness” on

to acknowledge the much broader range of locking script requirements, because not all unlocking scripts must contain signatures.

The UTXO is permanently recorded in the blockchain, and therefore is invariable and is unaffected by failed attempts to spend it bY reference in a new transaction .

Th script execution stack

Stack. A stack is a very simple data structure that can be visualized stack of cards.

allows two operations: push and pop.

Push adds an item can only act on the topmost item on the stack.

A stack data structure is also called a Last-In-First-Out, or “LIFO” queue.

Executes the script by processing each item from left to right.

Conditional operators evaluate a condition, producing a boolean result of TRUE or FALSE. A simple script

Ɔther nonzero value, or if the stack is empty after stack is FALSE (a zero-length empty value, noted as {1}) or if script execution is halted explicitly by an operator, such as OP\_VERIFY, execution is halted explicitly by an operator, such as vr\_” Figure 6-4. Bitcoin’s script validation doing simple math

Separate execution of unlocking and locking scripts

The | stack execution engine. If the unlocking script is executed without errors (e.g., it has no “dangling” operators left over), the main stack is copied and the locking script is executed.

### Pay-to-Public-Key-Hash (P2PKH)

Most applications would show the public key hash in hexadecimal encoding and not the familiar bitcoin address Base 58 Check format that \* begins with a “1.”

Combined validation script:

### Digital Signatures (ECDSA)

Elliptic Curve Digital Signature Algorithm, or ECDSA. ECDSA is the algorithm used for digital signatures

* The signature proves that the owner of the private key, who is by implication the owner of the funds, has authorized the spending of those funds.
* has authorized the spending of those funds. undeniable (nonrepudiation).
* Thirdly, the signature have not and cannot be modified by anyone after it has been signed.

Each transaction input is signed independently.

Neither the signatures nor the inputs have to belong to or be applied by the same “owners.”

“Coin Join” uses this fact to create multiparty transactions for privacy

Independent of any other input or signature. Multiple parties can collaborate to construct transactions and sign only one

Wikipedia’s Definition of a “Digital Signature

Believe that the message was created by a known sender (authentication), that the 1 sender cannot deny having sent the message (nonrepudiation), and that the message 1 was not altered in transit (integrity).

### How Digital Signatures Work

A mathematical scheme th at consists of two parts. The first part is an algorithm for creating a signature, using a private key (the signing key), from a message (the transaction). The second part is an algorithm that allows anyone to ver ify the signature given also the message and a public key

Creating a digital signature Sig = Fsig(Fhash(m), dA)

Sig = (R, 5)

Serialized into a bytestream using an international standard encoding scheme called the Distinguished stream using an internati

### Verifying the Signature

### Signature Hash Types (SIGHASH)

Signature implies a commitment by the signer to specific actions themselves. The signature implies a commitment by the signer to specific transaction data. In the simplest form, the signature applies to the entire transaction, thereby committing all the inputs, outputs, and other transaction

Of indicating which part of a transactions data is included in the hash signed by the private key using a SIGHASH flag. The

Every signature has a SIGHASH flag

SIGHASH\_ANYONECANPAY, which can be combined

Applied during signing and verification is that a copy of the The way SIGHASH flags are applied during signing and verification is that a copy v me transaction is made and certain fields within are truncated (set to Rigen of the serialized transaction and the result is hashed.

Software to construct and sign transpurpose bitcoin applications that enable novel uses.

### ECDSA Math

An ephemeral (temporary) private public key signa

Based on a random number k

Algorithm calculates the S value of the signature, such that: S=k’ (Hash(m) + dA \* R) mod p

Verification is the inverse of the signature generation function, using the R, S values

If the x coordinate of the calculated point P is equal to R, then the verifier can conIf the x coordinate of the calculate

### The Importance of Randomness in Signatures

Ephemeral private/public key pair.

If the same value K is used to create two signatures on different messages (transactions), then the signing private key can be calculated by anyone. Reuse of the same value for k in a signature algorithm leads to exposure of the private key!

Best practice is to not generate k with a random-number generator seeded with entropy, but instead to use a deterministic random process seeded with the transaction data itself. - RFC 6979

Different k for each transaction.

## Bitcoin Addresses, Balances, and Other Abstractions

When the blockchain explorer retrieved the transaction it also retrieved the previous transaction referenced in the input and extracted the first output from that older transaction.

The blockchain explorer extracted the locking script from each output, recognized it as a P2PKH script, and extracted the public key-hash from within. Finally, the blockchain explorer reencoded that

“Total Received” amount, the blockchain explorer first will decode the Base58Check encoding of the bitcoin address to retrieve the 160-bit hash of Bob’s public key that is encoded within the address.

Separate database of the outputs that are currently unspent, the UTXO set. To maintain this database, the blockchain explorer must monitor the bitcoin network, add newly created UTXO, and remove spent UTXO, in real time, as they appear in unconfirmed transactions. This

The most recent “strange transactions” that were not fully decoded: <https://block> [chain.info/strange-transactions](http://chain.info/strange-transactions).

# Advanced Transactions and Scripting

## Introduction

## Multisignature

At this time, standard multisignature scripts are limited to at most 3 listed public keys, meaning you can do anything from a 1-of-1 to a 3-of-3 multisignature or any keys, meaning you can do anything from a 1-of-1 to a 3-of-3 multisignature or any combination within that range. The limitation to 3 listed keys go my un time this book is published, so

Limit of 3 keys applies only to standard (also known as “bare”) multisignature scripts, not to multisignature scripts wrapped in a Pay-to-Script-Hash (P2SH)

General form of a locking script setting an M-of-N

M<Public Key 1><Public Key 2>…..<Public Key N> N CHECKMULTISIG

A bug in CHECKMULTISIG execution

First, CHECKMULTISIG pops the top item, which is N (in this example “3”). Then it pops First, CHECKMULTISIG pops the top item, which In this example, public keys A, B, and items, which are the public keys that can sign.

A bug in the implementation causes CHECKMULTISIG to pop one more item (M+1 total) than it should. The extra item is disregarded

Value must be present because if it is not present, when CHECKMULTISIG attempts to pop on an empty stack, it will cause a stack error and script failure (marking the transaction as invalid). Beca

Customarily O is used.

## Pay-to-Script-Hash (P2SH)

Offers corporate governance controls backup key. A multisignature scheme like that offers

Simple of that extra-large transaction would be borne by the customer in the form of fees.

P2SH was developed to resolve these practical difficulties and to make the use of P2SH was developed to resolve these practical difficulties

P2SH means “pay to a script matching this hash, a script that will be presented later this output is spent.”

In P2SH transactions, the locking script that is replaced by a hash is referred to as the In P2SH transactions, the locking script that is replaced by a hash time rather than as a locking script.

Shifts the burden in fees and complexity from the sender to the recipient (spender) of the den in fees

First, the redeem script is checked against the locking script to make sure the hash matches:

<2 PK1 PK2 PK3 PK4 PK5 5 CHECKMULTISIG> HASH160>redeem scriptHash> EQUAL

If the redeem script hash matches, the unlocking script is executed on its own, to unlock the redeem script:

<Sig1><Sig> 2 PK1 PK2 PK3 PK4 PK5 5 CHECKMULTISIG

### P2SH Address

Feature is the ability to encode a script hash as an address, as defined in BIP-13. PZS:

Base58Check encodings of the 20- Base58Check-encoded addresses that start with a “3.”

### Benefits of P2SH

Making the transaction smaller.

Sender’s wallet don’t need complex engineering to implement P2SH.

Burden of constructing the script to the recipient, not the sender.

Burden in data storage for the long script from the output (which P2SH shifts the burden in data storage for the long script for additionally to being stored

Shifts the burden in data storage for the long script from the present time (payment) to a future time (when it is spent).

Transaction fee cost of a long script from the sender to the recipient, who has to include the long redeem script to spend it.

### Redeem Script and Validation

not able to put a P2SH inside a P2SH redeem script, because the P2SH specification is not recursive.

RETURN during validation will cause the transaction to be marked invalid.

Includes the because you can lock bitcoin in a P2SH that cannot be spent later.

The P2SH locking script even if it corresponds to an invalid redeem script, because the script hash gives no indication of the script it represents.

P2SH transaction will be considered valid and accepted even if the redeem script is invalid.

## Data Recording Output (RETURN)

Cannot be spent, using the destination bitcoin address as a freeform 20-byte field. Because the address is used for data, it doesn’t correspond to a private key and the resulting UTXO can never be spent; it’s a fake payment. These transactions that can never be spent are therefore never removed from the These transactions that can UTXO database to forever increase, or “bloat.”

Version 0.9 of the Bitcoin Core client, a compromise was reached with the introduction of the RETURN operator. RETURN allows developers to add 80 bytes of nonpayment data to a transaction output. However, unlike the use of “fake” UTXO, the RETURN operator creates an explicitly provably unspendable output, which does not RETURN operator creates an explicitly

Be stored in the UTXO set. RETURN outputs are recorded on the DIOCкcnain, they consume disk space and contribute to the increase in the prow pool and burden full nodes with the cost of more expensive RAM.

Proof of Existence (<http://proofofexistence.com>) digital notarization service uses the 8-byte prefix DOC (<http://proofofexistence.com>) digital notarization Service uses Uie d-dyw p³

provably unspendable output

RETURN is usually an output with a zero bitcoin amount,

Two new coшaшu-ne options have been added in Bitcoin Core as of version 11 VIV. 1 default set to “1” to allow them. The option datacarriersize takes a numeric argument specifying the maximum size in bytes of the RETURN script, 83 bytes by default,

## Timelocks

Insactions or outputs that only allow spending after a point in time. Bitcoin has had a transaction-level timelock feature

New timelock features were introduced in late 2015 and mid-2016 that offer UTXO-level timelocks. These are CHECKLOCKTIMEVERIFY and CHECKSEQUENCEVERIFY.

### Transaction Locktime (nLocktime)

Relayed or included in the blockchain prior to the specified block height. If it is above 500 million, it is interpreted as a Unix Epoch timestamp (seconds since above 500 million, it is interpreted as a Unix Epoch timestamp (sec

Must be held by the originating syswith nLocktime specifying a future block or time must be held by the origin

Transmitted to the bitcoin network only after they become valid. Transaction will be rejected by the first node as invalid and will not be relayed to other nodes

#### Transaction locktime limitations

nLocktime has the limitation that while it makes it possible to spend some outputs in nLocktime has the limitation that while it makes it possible to spend som

Timelock restriction must be placed on the UTXO itself and be part of the locking script, rather than on the transaction.

### Check Lock Time Verify (CLTV)

BIP-65 - CHECKLOCKTIMEVERIFY (CLTV) was added to the scripting language. CLTV is a per output timelock, rather than a per-transaction timelock as is the case with nLocktime.

Takes one parameter as input, expressed as a number in the same format as nLocktime (either a block height or Unix epoch time). As indicated by the VERIFY suffix, CLTV is the type of opcode that halts execution of the script if the outcome is FALSE. If it results in TRUE, execution continues.

Invalid if

The stack is empty; or

2. The top item on the stack is less than 0: or “

3. The lock-time type (height versus timestamp) of the top stack item and the Lock time field are not the same; or

4. The top stack item is greater than the transaction’s nLocktime field; or

5. The nSequence field of the input is Oxffffffff.

### Relative Timelocks

BIP-68 and BIP-112 were activated in May 2016 as a

Both a transaction level feature and a script-level opcode.

Useful in bidirectional state channels and Lightning Networks

Allow a chain of two or more interdependent transactions to be held off chain, while imposing a time constraint on one

### Relative Timelocks with nSequence

#### Original meaning of nSequence

Never properly implemented) to allow modification of transactions in the mempool. In that

Modification of transactions in the mempool. In that use, a transaction that was not yet “finalized.” Such a transaction would be held in the mempool until it was replaced by another transaction spending the same inputs with a higher nSequence value

#### nSequence as a consensus-enforced relative timelock

Omic future developments. Transaction inputs with nSequence values less than 231 are interpreted as having a relative timelock.

The UTXO referenced in the input was mined. Since nSequence is a per-input field, a transaction may contain any number of timelocked inp (LU) PIVA inust timelocked inputs (nSequence &lt; 231) and inputs without a relative timelock (nSe quence &gt;= 23³¹).

Type-flag is used to differentiate between values counting blocks and values counting time in seconds. The type-flag is set in values counting blocks and values counting time in If the type-flag is set, then the nSe quence value is interpreted as a multiple of 512 seconds. If the type-flag is not set, the nSequence value is interpreted as a number of blocks.

When interpreting nSequence as a relative timelock, only the 16 least significant bits are considered.

### Relative Timelocks with CSV

The CSV opcode when evaluated in an UTXO’s redeem script allows spending only in a transaction whose input nSequence value is greater than or equal to the CSV parameter.

### Median-Time-Past

Everywhere. Everything is synchronized to create a common ledger. Bitcoin reaches consensus every 10 minutes about the state of the ledger as it existed in the past.

Proposed and activated at the same time as the Di COCKS. This is

Median-Time-Past is calculated by taking the timestamps of the last 11 blocks and finding the median. That median time then becomes consensus time and is used for all timelock calculations.

Changes the implementation of time calculations for nLocktime, CLTV, nSequence, and CSV. The consensus time calculated by Median-Time-Past is always approximately one hour behind wall clock time.

### TimeTimelock Defense Against Fee Sniping

Miners attempting to rewrite past blocks “snipe” higher-fee transactions from future blocks to maximize their profitability.

Have the option to pull transactions from the “present” into the rewritten “past” when they re-create block #100,000.

To prevent “fee sniping,” when Bitcoin C. creates transactions, it uses nLocktime to limit them to the “next block,” by default

Bitcoin Core sets the nLocktime on all new transactions to &lt;current block # + 1&gt; and sets the nSequence on all the inputs to OxFFFFFFFE to enable nLock time.

## Scripts with Flow Control (Conditional Clauses)

Conditional expressions can be “nested” indefinitely, meaning 1 that a conditional clause can contain another within it, which contains another, etc, Bitcoin Script flow control can be used to construct very complex scripts with hundreds or even thousands of possible execution paths.

Consensus rules impose a limit on the maximum size, in bytes, of a script.

Control using the IF, ELSE, ENDIF, and NOTIF opcodes. Additionally, conditional expressions can contain boolean bitcoin operators such as BOOLAND, BOOLOR, and NOT.

Logical condition comes before the IF, which makes it look “backward,” like this:

### Conditional Clauses with VERIFY Opcodes

Any {opcode that ends in VERIFY.

The VERIFY suffix means that if the condition evaluated is not TRUE, execution of the script terminates immediately and the transaction is deemed invalid.

A guard clause, continuing only if a precondition is met.

Attach a precondition (guard clause), then VERIFY is better. If, however, we want to have more than one execution path (flow control), then we need an If\_ELSE

Opcodes that end in VERIFY do not leave the result on the the stack.

### Using Flow Control in Scripts

Construct a redeem script that offers multiple execution paths, each a different way of redeeming the UTXO.

To spend, we construct an unlocking script that navigates the execution path by putting the appropriate TRUE and FALSE values on the stack at each flow control point.

Why FALSE TRUE? Isn’t that backward?

## Complex Script Example

Because the two values are Why FALSE TRUE? Isn't that backward? Because the two values art second. TRUE is therefore popped first by the first IF opcode.

## Segregated Witness

In cryptography, the term “witness” is used to describe a solution to a cryptographic be a cryptography, the term “witness’ ‘witness” is used to descr zzle. In bitcoin terms, the witness satisfies a cryptographic condition place unspent transaction output (UTXO).

Unlock that UTXO for spending.

Data was embedded in the transaction as part of each input. The term segregated witness, or segwit for short, simply means separating the signature or unlocking script of a specific output. Think “separate scriptSig,” or ‘separate signature” in the simplest form.

Clients may request transaction data with or without the accompanying witness data.

### Why Segregated Witness?

#### Transaction Malleability

Since the witness data is the only part of the transaction that can be modified by a third party (see “Transaction identifiers” on page 181), removing it also removes the opportunity for transaction malleability attacks.

Improves the implementation of many other protocols that rely on advanced bitcoin transaction construction, such as payment channels, chained transactions, and lightning networks.

#### Script Versioning

Locking script is preceded bY a script version number, similar to how transactions and blocks have ceded by a script

Upgraded in a backward-compatible way

Accelerate the rate of innovation in bitcoin.

#### Network and Storage Scaling

In some cases these scripts account for the majority (more than 75%) of the data in a transaction. By moving the witness data outside the transaction, Segre data in a transaction. By moving the witness data outside the trans 0 1 after validating the signatures, or ignore it altogether

#### Signature Verification Optimization

Reduce the algorithm’s computational complexity.

Data-hashing computations increased in O(n?) with respect to the number of signature operations, segwit, the algorithm is changed to reduce the complexity to O(n).

#### Offline Signing Improvement

Amount of Qata part of the commitment hash that is signed, an offline device does not need the previous transactions.

### How Segregated Witness Works

A change to how individual UTXO are spent and therefore is a per-output feature. -

Can spend Segregated Witness outputs or traditional (inline-witness) outputs or both.

Segregated Witness UTXO, however, specifies a locking script that can be satisfied with witness data outside of the input segregated).

### Soft Fork (Backward Compatibility)

To an old wallet or node, a Segregated Witness output looks like an output that anyone can spend. Such outputs can be spent with an empty signature, therefore the fact that there is no signature inside the transaction (it is segregated) does not invalidate the transaction.

### Segregated Witness Output and Transaction Examples

#### Pay-to-Witness-Public-Key-Hash (P2WPKH)

Segregated Witness outputs locking script is much simpler than a traditional output.

Does not require a signature ས་་ Ųསྐྱུ 1 is interpreted as a version number (the witness version) and the second part (20 bytes) the equivalent of a locking script known as a witness program. The 20-byte witness as in a P2PKH script program is simply the hash of the public key, as in a P2PKH script

An empty scriptSig and includes a Segon that input. Instead, Bob’s transaction nas an

#### Wallet construction of P2WPKH

P2WPKH should only be created by the pavee (recipient) and not converted by the sender from a known public key, p2pKH script, (recipient)

Outputs must be constructed from the hash of a compressed public key.

#### Pay-to-Witness-Script-Hash (P2WSH)

P2SH script references the hash of a redeem script that defines a 2-of-3 multi signature requirement to spend funds.

The redeem script (whose hash matches the script hash in the P2SH output) and the signatures necessary to satisfy that redeem script, all inside the transaction input:

Simpler and omits the various script operands that You see in P2SH scripts. Instead, the Segregated Witness program consists of two values pushed to the stack: a witness version (O) and the 32-byte SHA256 hash of the redeem script.

Difference in the selection of the hashing algorithm is deliberate and is used to differentiate between the two types of witness programs (P2WPKH and P2WSH) by the length of the hash and to provide stronger security to P2WSH (128 bits of security in P2WSH versus 80 bits of security in P2SH).

#### Differentiating between P2WPKH and P2WSH

They look very similar, but are interpreted very differently: one is interpreted as a public key hash, which is satisfied by a signature and the other as a script hash, which is satisfied by a redeem script.

Length of the hash:

* The public key hash in P2WPKH is 20 bytes
* The script hash in P2WSH is 32 bytes

### Upgrading to Segregated Witness

Implemented as a backward-compatible upgrade, old and new clients can coexist.

Ability of a sender’s wallet that is not segwit-aware to make a payment to a recipient’s wallet that can process segwit transactions

Ability of a sender’s wallet that is segwit-aware to recognize and distinguish between recipients that are segwit-aware and ones that are not, by their addresses.

Embedding Segregated Witness inside P2SH

Use segwit to reduce transaction fees, taking advantage of the discount that applies to witness data.

#### Pay-to-Witness-Public-Key-Hash inside Pay-to-Script-Hash

To make it possible for any client to pay his company, regardless of whether their wallets are upgraded for segwit, Mohammed’s wallet can embed the P2WSH witness program inside a P2SH script.

First, Mohammed’s wallet hashes the redeem script with SHA256 (just once). Let’s use bx to do that on the command-line:

Next, the hashed redeem script is turned into a P2WSH witness program:

Witness program itself is hashed with SHA256 and RIPEMD160,

The wallet constructs a P2SH bitcoin address from this hash.

Lock the output

#### Segregated Witness addresses

The native segwit address format is defined in BIP-173:

Only encodes witness (P2WPKH and P2WSH) scripts. It is not compatible BIP-173 only encodes witness (P2WPKH and P2WSH) scripts. It is not compa

Also called bech32 addresses, use of a “BCH” error detection algorithm and 32-character encoding set.

32 lower-case-only alphanumeric character

Error detection algorithm is a vast improvement over the previous checksum algorithm (from Base58Check), allowing not only detection but also correction sum algori

A segwit bech32 string is up to 90 characters long and consists of three parts:

The human readable part This prefix “bc” or “tb” identifying mainnet or testnet.

The separator

The digit “1”, which is not part of the 32-character encoding set and can only appear in this position as a separator separator

The data part A minimum of 6 alphanumeric characters, the checksum encoded witness script

#### Transaction identifiers

Before segwit, transactions could have their signatures subtly modified by third parBefore segwit, transactions could have their signatures subtly modified by third parties, changing their transaction ID (hash) without changing any fundamental properties (inputs, outputs, amounts).

This created opportunities for denial-of-service attacks as well as attacks against poorly written wallet software that assumed unconfirmed transaction hashes were immutable.

Transactions have two identifiers, txid With the introduction of Segregated Witness, transactions have two identifiers, txid t and wtxid. The traditional transaction ID txid is the double-SHA256 hash of the and wtxid serialized transaction, without the witness data

Wtxid is the doubleSHA256 hash of the new serialization format of the transaction with witness data

Segwit transaction has empty scriptSigs in every input, there is no part of the transaction that can be modified by a third party. Therefore, in segwit transaction, the txid is immutable by a third party, even when the transaca segwit transaction, the

Wtxid should be considered malleable until the trans- 1 action is confirmed.

Txid of a segwit transaction can be considered immutable by third parties and only if all the inputs of the transaction are segwit inputs.

### Segregated Witness’ New Signing Algorithm

Signatures in bitcoin transactions are applied on a commitment hash,

Address this problem bY changing the way the Segwit represented an opportunity

New algorithm achieves two important goals. Firstly; the number of hash operations increases by a much more gradual O(n) to the number of signature operations,

Reducing the opportunity to create denial-of-service attacks with overly complex transactions. Secondly, the commitment hash now also includes the value (amounts) transactions. Secondly, the commitment nash now also includes the value (amounts) specific input value without needing to “fetch” and check the previous transaction

#### Economic Incentives for Segregated Witness

Nonmining full nodes are not compensated, so they incur these costs because they have a need to run an authoritative fully validating full-index

The calculation of fees based on transaction size treats all the data in the transaction equal in cost. But

Disk Space’

Pruning” older transactions.

1 CPU

Validated, which requires CPU time.

Bandwidth Every transaction is transmitted (through flood propagation) across the network least once.

Memory

Nodes that validate transactions keep the UTXO index or the entire UTXO set in memory to speed up validation.

Growth of the UTXO set contributes disproportionately to the cost of running a node.

The most expensive part of a transaction are the newly created outputs, as

Witness data are only validated once and then never used again.

Wallets must implement some strategy for assembling transactions that takes into consideration a number of factors, such as privacy (reducing address reuse), frag: mentation (making lots of loose change), and fees. If th

Transactions consume UTXO in their inputs and create new UTXO with their out

Let’s consider the difference between inputs and outputs and call that the “Net-new-UTXO.”

Both transactions are less expensive when segregated witness is implemented. But comparing the costs between the two transactions, we see that before Segregated Witness, the fee is higher for the transaction that has a negative Net-new-UTXO. After Segregated Witness, the transaction fees align with the incentive to minimize new segregated Witness, the transaction fees align TXO creation by not inadvertently penalizing transactions with many inputs.

Segwit reduces the overall cost of transactions by discounting witness data and increasing the capacity of the bitcoin blockchain. Secondly, segwit’s discount on witness data corrects a misalignment of incentives that may have inadvertently created more bloat in the UTXO set.

# The Bitcoin Network

## Peer-to-Peer Network Architecture

Architecture on top of the internet. Term peer-to-peer, or P2r, means that the computers that participate in the net

There is no server, no centralized service, and no hierarchy within the network.

Reciprocity acting as the incentive for participation. P2P networks are inherently resilient, decentralized, and open. 16

Decentralization of control is a core design principle that can only be achieved and maintained by a flat, decentralized p2p consensus network.

The term “bitcoin network” refers to the collection of nodes running the bitcoin P2P protocol. In addition to the bitcoin P2P protocol, there are other protocols such as stratum that are used for mining and lightweight or mobile wallets. These additional protocols are provided by gateway routing servers

‘Extended bitcoin network” to refer to the overall network that includes the bitcoin P2P protocol, pool-mining protocols, the Stratum protocol, and any other related protocols connecting the components of

## Node Types and Roles

Functions: routing, the blockchain database, mining, and wallet services.

All nodes validate and propagate transactions and blocks, discover and maintain connections to peers.

Full nodes can autonomously and authoritatively verify any transaction without external reference.

A method called simplified payment verification, or SPV nodes or lightweight nodes

## The Extended Bitcoin Network

Listening nodes running various versions of the bitcoin reference cli(Bitcoin Core) and a few hundred nodes running various other implementations

Act as network edge routers, allowing various other services (exchanges, wallets, block explorers, merchant payment processing) to be built on top.

A number of pool servers and protocol gateways that connect nodes running other protocols.

Figure 8-2. Different types of nodes on the extended bitcoin network

Reference Client (Bitcoin Core)

##### Full Block Chain Node

##### 

##### Solo Miner

##### 

##### Lightweight (SPV) wallet

##### 

##### Pool Protocol Servers

##### 

##### Mining Nodes

##### 

##### Lightweight (SPV) Stratum wallet

Figure 8-3. The extended bitcoin network showing various node types, gateways, and protocols

Bitcoin Relay Networks

Miners must minimize the time between the propagation of winning block and the beginning of the next round of competition. In mining, network latency is directly related to profit margins.

Bitcoin Relay Network created by core developer Matt Corallo in 2015 to enable fast synchronization of blocks between miners with very low latency.

The original Bitcoin Relay Network was replaced in 2016 with the introduction of u develoner Matt Corallo. FIBRE is a UDP-based relay network that relays blocks core developer Matt Corallo FIBRE implements compact block optimization to further reduce the amount of data transmitted and the network latency.

Falcon uss “cut-through 1 - routing” insta o “stor-an-orwar” to ruc latency by propagating parts o

Not replacements or bitcoin’s P2P network. Insta thy ar overlay networks that provide additional connectivity between nodes

## Network Discovery

The geographic location of other nodes is irrelevant; the bitcoin network topology is not geographically defined.

To connect to a known peer, nodes establish a TCP connection, usually to port 8333

Nversion

nLocalServices currently just NODE\_NETWORK

nTime current time

addrYou IP address of the remote node as seen from this node

addrMe The IP address of the local node, as discovered by the local node

Subver sub-version showing the type of software running on this node

BestHeight The block height of this node’s blockchain

Establish a connection by sending a verack.

The first method is to query DNS using a number of “DNS seeds,” which are DNS servers that provide a list of IP addresses of bitcoin nodes.

Custom implementations of BIND (Berkeley Internet Name Daemon) that return a random subset from a list of bitcoin node addresses collected by a crawler or a long-running bitcoin node.

Seeds is controlled by the option switch -dnsseed (set to 1 by default,

A bootstrapping node that knows nothing of the network must be giver | the IP address of at least one bitcoin node,

Command-line argument -seednode can be used through further introductions. The command-line argument

Newly connected node can send getaddr to the neighbors, asking them to return a list of IP addresses of

A node must connect to a few different peers in order to establish diverse paths into bitcoin network. Paths are not reliable-nodes come and go-and

Only one connection is needed to bootstrap, becau..

Unnecessary and wasteful of network resources to connect to than a handful of nodes.

Can list the peer connections with the command getpeerinfo:

If a node has not communicated on a connection for more than 90 minutes, it is assumed to be disconnected and a new peer will be sought

Network dynamically adjusts organically grow and shrink as needed

## Full Nodes

Network to receive updates about new hin-i-c The full blockchain node relies on the network to ¨¨чew blocks ¸¨¨“w blocks cw blockchain.

It’s easy to tell if you’re running a full node because it requires more than one hundred gigabytes of persistent storage (disk space) to store the full blockchain. If

Most common implementation is the reference client Bitcoin Core, also known as the Satoshi client

Identified as “Satoshi” in the sub-version string sent in the version message and shown by the command getpeerinfo as

## Exchanging “Inventory”

The genesis block, which is statically embedded in the client software.

The version message, because that contains BestHeight, a node’s current blockchain height (number of blocks).

Peered nodes will exchange a getblocks message that contains the hash (fingerprint) of the nodes will exchange a getblocks message

The longer blockchain has more blocks than the other node and can identify which blocks the other node needs in order to “catch up.” It will identify the first 500 blocks to share and transmit their hashes using an inv (inventory) message.

Node missing these blocks will then retrieve them, by issuing a series of getdata

It will start requesting blocks from all of its connected peers, spreading the load and ensuring that it doesn’t overwhelm any peer with requests. The node keeps track of how many blocks are “in transit” per peer connection,

(MAX\_BLOCKS\_IN\_TRANSIT\_PER\_PEER).

Request new ones as previous requests are fulfilled,

This process of comparing the local blockchain with the peers and retrieving any

It starts by sending getblocks, gets an inv response, and starts downloading the missing blocks.

## Simplified Payment Verification (SPV) Nodes

Method is used to allow them to operate without storing the full blockchain.

Download only the block headers and do not download the transactions SPV nodes download only

Resulting chain of blocks, without transactions, is 1,000 times smaller than the full blockchain.

Relies on peers to provide partial views of relevant parts of the blockchain on demand.

Verifies transactions by reference to their depth in the blockchain instead of their height.

Idity of the transaction by confirming that the UTXO remains unspent. Will establish a link between the transaction and the block that contains it, using a establish

Waits until it sees path (see “Merkle Trees” on page 217). Then, the SPV node waits until it sees

Proof, by proxy, that the transaction was not a double-spend.

## Bloom Filters

Probabilistic search filter, a way to describe a desired pattern without specifying it exactly.

Nodes to ask their peers for transactions matching a specific pattern, without revealing exactly which addresses, keys, or transactions they are searching for.

### How Bloom Filters Work

Implemented as a variable-size array of N binary digits (a bit field) Bloom filters are implemented as a variable-size

Choosing different length (N) bloom filters and a differresults for a specific input. By choosing different length (N) bloom accuracy and therefore privacy.

As a bloom filter is filled with more patterns, a hash function result might coincide with a bit that is already set to 1, in which case the bit is not changed.

More bits set to 1 and the accuracy of the filter decreases. Are set to 1, then the pattern is probably recorded in the bloom filter. Because the bits may be set because of overlap answer is not certain, but is rather probabilistic.

In simple terms, match is a “Maybe, Yes.”

Any one of the bits is set to 0, this proves that the pattern was not recorded in the bloom filter. A negative match is “Definitely not!”

## How SPV Nodes Use Bloom Filters

SPV node will initialize a bloom filter as “empty”; in that state the bloom filter will not match any patterns.

1 then make a list of all the addresses, keys, hashes that it is interested in

* By extracting the public key hash and ript hash and transaction IDs
* Send a filterload message to the peer, containing the filter to use on the connection.
* Full node checks several parts o

Will send a merkleblock message containing block headers for blocks matching the filter and a merkle path for each matching transaction:

Also send tx messages containing the transactions matched by the filter.

SPV node discards any false positives and uses the correctly matched transactions to update its UTXO set and wallet balance.

Interactively add patterns

It is not possible to remove a pattern from a bloom filter, a node to clear and resend a new bloom filter if a pattern is no longer desired. Send a filterclear

BIP-37 (Peer Services)

## SPV Nodes and Privacy

Reveals no information about whether it is using some all transactions and

With bloom filters, an adversary monitoring the traffic of an SPV SPV client or connected to it directly as a node in the P2P network can collect enough information over time to learn the addresses in the wallet

## Encrypted and Authenticated Connections

Two solutions that provide encryption of the communications: Tor Transport and P2P Authentication and Encryption with BIP-150/151.

### Tor Transport

The Onion Routing network, is a software project and network offers encryption and encapsulation of data through randomized network paths. That offer anonymity, untraceability and privacy.

As of Bitcoin Core version 0.12, a node will offer a hidden Tor service automatically if

### Peer-to-Peer Authentication and Encryption

BIP-150 offers optional peer authentication that allows nodes to authenticate each other’s identity using ECDSA and private keys.

requires that prior to authentication the two nodes have established encrypted communications

### Transaction Pools

Network maintains a temporary list of unconfirmed transactions called the memory pool, mempool, or transaction pool.

Known to the network but are not Yet pool to keep track of transac

Relayed to the neighboring nodes to propagate on the network.

Separate pool of orphaned transactions.

Cendants are found. Through this process, the arrival of a parent transactON UR uniting the orphans with their parents all the way down the chain.

“UTXO ›ool” sounds similar to the transaction pool, it represents a different set of data.

UTXO pool is not initialized empty but instead contains millions of entries of unspent transaction outputs, everything that is inspent from all the way back to the genesis block. T

Housed local memory or as an indexed database table on persistent storage.

-་ UTXO pool represents the emergent consensus of the network and therefore will vary little between nodes.

# The Blockchain

## Introduction

An ordered, back-linked list of block of transactions.

Be stored as a flat file, or ina simple database.

Metadata using Google’s LevelDB database.

Previous block, known as the parent block, through the “previous block hash” field in block header.

First block ever created, known as the genesis block.

One parent, it can temporarily have multiple children.

“Fork,” a temporary situation that occurs when different blocks are discovered almost simultaneously by different miners (see

Each block can have only one parent.

After 100 blocks back there is so much stability that the coinbase transaction-the transaction containing newly mined bitcoin-can be spent.

While the protocol always allows a chain to be undone by a longer chain and while the possibility of any block being reversed always exists

## Structure of a Block

The block header is 80 bytes, whereas the average transaction is at least 400 bytes

Table 9-1. The structure of a block

## Block Header

Consists of three sets of block metadata. 3

Za reference to a previous block hash, w

The difficulty, timestamp, and nonce, relate

The merkle tree root, a summary of all transactions in the block

## Block Identifiers: Block Header Hash and Block Height

Hashing the block header twice through the SHA256 algorithm.

Block hash - more accurately - block header hash because Only the block header is used to compute

Uniquely and unambiguously and can be independently verified by any no by simply hashing the block header

Block hash is not actually included inside the block’s data structure, it Is computed by each node as the block is received from the network.

Block height is also not a part of the block’s data structure; it is not stored within the block. Each node dynamically identifies a block’s position (height) in the blockchain when it is received from the bitcoin network.

## The Genesis Block

A hidden message within it. The coinbase transaction input contains the text “The Times 03/Jan/2009 Chancellor on brink of second bailout for banks.”

## Linking Blocks in the Blockchain

The local copy of the blockchain is constantly updated as new blocks are found validate these blocks and then link them to the existing blockchain. To establish link, a node will examine the incoming block header and look for the “previous block hash.”

## Merkle Trees

A binary hash tree, is a data structure used for efficiently summarizing and verifying the integrity of large sets of data.

Describe a branching data structure, but these

Root” at the top and the “leaves” at the bottom of a diagram, as you will see

Figure 9-1. Blocks linked in a chain by reference to the previous block header hash

Constructed by recursively hashing pairs of nodes until there is only one hash, called the root or **merkle root.**

2\*log (N) calculations, making this a very efficient data structure.

Constructed bottom-up.

Transactions are not stored in the merkle tree; rather, their data is hashed and the resulting hash is stored in each **leaf node** as HA, HB, HC, and HĎ:

На = SHA256(SHA256(Transaction A) &gt;

HAB = SHA256(SHA256(HA + HB &gt; &gt;

If there is an odd number of transactions to summarize, the last transaction hash will be duplicated to create an even number of leaf nodes, also known as a **balanced tree.**

An authentication path or merkle path connecting the specific transaction to the root of the tree.

Paths of 10 or 12 hashes (320-384 bytes), which can provide proof of a single transaction out of more than a thousand transactions in a megabyte-sized block.

## Merkle Trees and Simplified Payment Verification (SPV)

SPV node that is interested in incoming payments to

Will establish a bloom filter (see

WHen a peer sees a transactions tion that matches the bloom filter, it will send that block using a merkleblock mention that matches the bloom filter, it will send that block using a merkleblock mes

More than a thousand times less than a full block

## Bitcoin’s Test Blockchains

Mainnet. There are other bitcoin blockchains that are used for testing purposes: at this time testnet, segnet, and regtest. Let’s look at each in turn.

### Testnet-Bitcoin’s Testing Playground

### Segnet-The Segregated Witness Testnet

### Regtest-The Local Blockchain

Using Test Blockchains for Development

Test your code locally on a regtest as you develop it. Once you are ready to try it on a public netlocally on a regtest as you develop it. Once you are ready to try it on a public net diversity of code and applications.

Works as expected, switch to mainnet to deploy it in production.

# Mining and Consensus

## Introduction

Mechanism that underpins the decentralized learinghouse, by which transactions are validated and cleared.

An incentive scheme that aligns the actions of miners with the security of the network, while simultaneously implementing the monetary suppl.

Based on a

Process is called mining because the reward (new coin generation) is designed to simulate diminishing returns, just like mining for precious metals.

### Bitcoin Economics and Currency Creation

New coins decreases like this exponentially over 32 “halvings” until block 720,000 (mined approximately in year 2137), when it reaches the minimum currency unit of 1 satoshi.

## Decentralized Consensus

Satoshi Nakamoto’s main invention is the decentralized mechanism for emergent consensus. Emergent, because consensus is not achieved explicitly there is no election sensus. Emergen, because consensus is not

Asynchronous interaction of thousands of independent nodes, all following of the asynch

Security model that does not depend on central authority or trust, derive this invention.

## Independent Verification of Transactions

Every bitcoin node that receives a transaction will first verify the transaction. Standardness rules

Against a long checklist of criteria:

* Syntax and data structure
* Neither lists of inputs or outputs are empty.
* Size in bytes is less than MAX\_BLOCK\_SIZE. Greater than or equal to 100 bytes
* Each output value, as well as the total, must be within the allowed range of values (less than 21m coins, more than the dust threshold). Also, all inputs(less than 21m; coins, more than O).
* None of the inputs have hash=0, N=-1 - (coinbase transactions should not be relayed).
* nLocktime is equal to INT\_MAX, or nLocktime and nSequence values are satisfied according to Median TimePast.
* Number of signature operations (SIGOPS) contained in the transaction is less than the signature operation limit.
* Unlocking script (scriptSig) can only push numbers on the stack, and the locking script (scriptPubkey) must match IsStandard forms
* A matching transaction in the pool, or in a block in the main branch, must exist.
* If the referenced output exists in any other transaction in the pool, the transaction must be rejected.
* Branch and the transaction pool to find the referenced output transaction is missing for any p
* COINBASE\_MATURITY (100) confirmations.
* Referenced output must exist and cannot already be spent.
* Reject if the sum of input values is less than sum of output values.
* Reject if transaction fee would be too low (m \nRelayTxFee)
* Unlocking scripts for each input must validate against the corresponding output locking scripts.
  + Conditions can be seen in detail in the functions AcceptToMemoryPool, Check Transaction, and CheckInputs in Bitcoin Core.

Every node builds a pool of valid (but unconfirmed) transactions known as the Transaction pool, memory pool, or mempool.

## Mining Nodes

Arrival of a new block has special significance means someone else won

## Aggregating Transactions into Blocks

Aggregate these transactions node. Unlike other nodes.

Upon receiving i block 277,315 and validating it, Jing’s node will also compare it against all the transactions in the memory pool and remove any that were included in bloc k

### The Coinbase Transaction

Amount of reward that Jing collects for mining block is the sum of the coinbase reward (25 new bitcoin) and the transaction fees

### Coinbase Reward and Fees

First calculates the total amount of transaction fees by adding all the inputs and outputs of the 418 transactions that were added to the block.

The fees are calculated as: Total Fees = Sum(Inputs)- Sum(Outputs)

Based on the block height, function GetBlockSubsidy

COIN constant 000,000 satoshis). This sets the initial reward (nSubsidy) at 5 billion satoshis.

The function calculates the number of halvings that have occurred by dividing, current block height by the halving interval

Maximum number of halvings allowed is 64, so the

Uses the binary-right-shift operator to divide the reward. Used because it is more efficient than multiple repeated divisions. To avoid a potential bug, the shift operation is skipped after 63 halvings, and the subsidy is set to 0.

### Structure of the Coinbase Transaction

Table 10-1. The structure of a “normal” transaction input

Table 10-2. The structure of a coinbase transaction input

* Transaction Hash All bits are zero: Not
* Output Index All bits are ones: 0xFFFFFFFF Coinbase Data Arbitrary data used for extra nonce and mining tags. In v2 blocks; must begin with block height

### Coinbase Data

Do not have an unlocking script

‘Must be between 2 and 100 bytes, can be used by miners in any way thy want; it is arbitrary data.

Include extra nonce values and strings identifying the mining pool.

Per BIP-34, version-2 blocks (blocks with the version field set to 2) must contain block height index as a script “push” operation

The first byte, 03, instructs the script execution engine to push the next three bytes onto the script stack (see Table B-1). The next three bytes, 0x443b04, are the block Height encoded in little-endian format (backward, least-significant byte first). Reverse the order of the bytes and the result is 0x043b44, which is 277,316 in decimal

Final part of the coinbase data (2f503253482f) is the ASCII-encoded ‘string /P2SH/, which indicates that the mining node that mined this block supports string /P2n, which indicates that the mining

Example 10-6. Extract the coinbase ata rom the genesis block

We compile the code with the GNU C++ compiler and run the resulting executable

## Constructing the Block Header

Table 10-3. The structure of the block header

Version number which is encoded in little-endian format in bytes as

“Previous Block Hash” (also known as prevhash). Parent

Summarize all the transactions with a merkle tree,

The coinbase transaction is listed as the first transaction in the block.

2nd add a 4-byte timestamp. Encoded as a Unix “epoch” timestamp, which is based on the number of seconds elapsed from January 1, 1970, midnight UTC/GMT.

The target, which defines the required Proof-of-Work

Which defines the required Proof-of-Work to make mantissa-exponent encoding of the targe

A 1-byte exponent, followed by a 3-byte mantissa (coefficient).

The nonce, which is initialized to zero.

The goal is now to find a value for the nonce that results in a block header hash that is less than the target.

Test billions or trillions of nonce values

## Mining the Block

Way to produce a hash result matching a specific target is to try again and again, randomly modifying the input until the desired hash result appears by chance.

### Proof-of-Work Algorithm Target Representation

An arbitrary-length data input and produces a fixed-length deterministic result, a digital fingerprint of the input For

The hash or digest of the phrase and depends on every part of the phrase. Adding a single letter, punctuation mark, or any other character will produce different hash \_

Make a challenge out of this algorithm, let’s set a target: find a phrase that pro duces a hexadecimal hash that starts with a zero.

If the output of the hash functions evenly distributed we would expect to find a result with a O as the hexadecimal prefix once every 16 hashes (one out of 16 hexadecimal digits O through’ F).

From the perspective of an observer who knows that the target of the dice game is 2, if someone has succeeded in casting a winning throw it can be assumed that they attempted, on average, 36 throws. In other

When the algorithm is a based on a deterministic function such as SHA256, the input itself consti-| proof that a certain amount of work was done to produce a result below the target. Hence, Proof-of- Work.

A lower target means it is more difficult to find a hash below the target. The target and difficulty are inversely related.

Set the desired difficulty (in bits, how many of the leading bits must be zero) and

Increasing the difficulty by 1 bit causes a doubling in the time it takes you can see, increasing the difficulty by I bit causes a mig i ɩne time it takes strain one more bit to zero, You decrease the search space by half.

### Target Representation

The formula to calculate the difficulty target from this representation is:

Target = coefficient \* 2(8\*(exponent-3))

### Retargeting to Adjust Difficulty

Every 2,016 blocks, all nodes retarget the Proof-of-Work.

Measures the time it took to find the last 2,016 blocks and compares that to the expected time of 20,160 took to find the last 2,016 blocks and compares that to the expected time of 20,16 actual timespan and desired timespan is calculated and a proportionate adjustment (up or down) is made to the target.

If the network is finding (up or down) is made to the target. In simple terms: If the network 1 discovery is slower than expected, the difficulty decreases (target increases).

New Target = Old Target \* (Actual Time of Last 2016 Blocks / 20160 minutes)

A retargeting bias toward higher difficulty by 0.05%. An off-by-one error in the original Bitcoin Core client it uses 2015

To avoid extreme volatility in the difficulty, the retargeting adjustment must be less than a factor of four (4) per cycle.

Target is independent of the number of transactions or the value of transaction -s. This means that the amount of hashing power and therefore electricity | expended to secure bitcoin is also entirely independent of the number of transactions.

Enough hashing power is under the control of miners acting honestly in pursuit of the reward, it is enough to prevent “takeover” attacks and, therefore, it is enough to secure bitcoin.

Primary influence on the mining market is the price of one kilowatthour of electricity in bitcoin, because that determines the profitability of mining and

## Successfully Mining the Block

Nonce is only 32 bits, after exhausting all the nonce possibilities (about 4 billion), the mining hardware changes the block header (adjusting the coinbase extra nonce space or timestamp) and resets the nonce counter, testing new combinations.

## Validating a New Block

Each node performs a series of tests to validate it before propagating it

Functions CheckBlock and CheckBlock Header and include:

* Syntactically valid ‘
* Less than the target (enforces the Proof-of- Work)
* Timestamp is less than two hours in the future (allowing for time DIC size is within acceptable limits
* First transaction (and only the first) is a coinbase transaction
* All transactions within the block are valid

## Assembling and Selecting Chains of Blocks

Validated a new block, it will then attempt to assemble a chain by connect the block to the existing blockchain.

Nodes maintain three sets of blocks:

* those connected to the main blockchain
* those that form branches off the main blockchain (secondary chains)
* blocks that do not have a known parent in the known chains (orphans)

Invalid blocks are rejected as soon as any one of the validation criteria fails

Main chain” at any time is whichever valid chain of blocks has the most cumulaProof-of-Work associated with it.

Secondary chains occur as a result of an almost simultaneous mining of blocks at the same height.

Most of the time, the parent will be the “tip” of the main chain, meaning this new block extends the main chain. For

Node will attach the new block to the secondary chain it extends and then compare the work of the secondary chain to secondary cha

If the secondary chain has more cumulative work, node will select secondary chain as its new main chain, making the old main chain a secondary chain.

* A miner, it will now construct a block extending this new, longer, chain.

Orphan blocks usually occur when two blocks that were mined within a short time of each other are received in reverse order (child before parent).

### Blockchain Forks

occur naturally as a result of transmission delays in the global network. We will also look at deliberately induced forks later

Figure 10-3. Visualization of a blockchain fork event: two blocks found simultaneously

Both blocks are valid, both blocks contain a valid solution to the Proof-of-Work, and both blocks extend the same parent (block “star”).

Since it main chain. Later, Node X also received the “upside-down angle IVR Since it LI v was received so I assumed to have “lost” the race. Yet, the “upside-down tri- it is angle” block

Neither side is “correct,” or “incorrect.” Both are valid perspectives of the blockchain. Only in hindsight will one prevail,

Voting with their hashing power. Their supports the chain that they have elected as the main chain.

Forks are almost always resolved within one block.

more cumulative work than the other chain

Transactions are Re-inserted in the mempool for inclusion in the next block to become a part of the main chain.

A one-block fork might occur every day, a two-block fork occurs at most once every few weeks.

Interval of 10 minutes is a design compromise between fast confirmation times (settlement of transactions) and the probability of a fork.

* A slower block time would decrease the number of forks but make settlement whereas

## Mining and the Hashing Race

Introduction of ASIC mining lead to another giant leap in mining power, by placing the SHA256 function directly on silicon chips speci alized for the purpose of mining.

Chips could deliver more power in a single box than the entire bitcoin network in 2010.

No longer about how much mining can be done with one chip, but how many chips can be squeezed into a building, while still dissipating the heat and providing adequate power.

### The Extra Nonce Solution

Bitcoin mining has evolved to resolve a fundamental limitation in the Since 2012, bitcoin mining

Culty increased, miners often cycled through all 4 billion values of the nonce without finding a block. However, this was easily resolved by updating the block timestamp to account for the elapsed time.

Allow miners to iterate through the values of the nonce again with different results. Once mining hardware exceeded 4 GH/sec, however, this approach

The coinbase script can store between 2 and 100 bytes of data, nonce values.

included in the merkle tree, means that any change in the coinbase script causes the merkle root to change.

Eight bytes of extra nonce, plus the 4 bytes of “standard” nonce allow miners to explore a total 2^96 (8 followed by 28 zeros) possibilities per second without having to modify the timestamp

### Mining Pools

Participating in a pool, miners get a smaller share of the overall reward, but typically get rewarded every day, reducing uncertainty. He might find two blocks in 4 years and make a very large profit. Or he might not find a block for 5 years and suffer a bigger financial loss. Even worse, the difficulty of the bitcoin Proof-of-Work algorithm is likely to go up significantly over that period, at the current rate of growth of hashing power, meaning the miner has, at most, one year to break even before the hardware is effectively obsolete and must be replaced by more powerful mining hardware.

Pool sets a higher target (lower difficulty) for earning a share, typically more than 1,000 times easier the bitcoin network’s target. When someone in the pool successfully mines a the bitcoin network’s target. When someone in the pool successfuу to the number of shares they contributed to the effort.

Managed pools

The owner of the pool server is called the poo! Operator, and charges pool miners a percentage fee of the earnings.

And a pool-mining protocol that coordinate the activities

Allows the pool server to validate blocks and transactions on behalf of the pool min

Bitcoin software running on the full node needs to be monitored, maintained, and upgraded frequently. Any downtime caused by a lack of maintenance or lack of resources will hurt the miner’s profitability.

For many miners, the ability to mine without running a full node is another big benefit of joining a managed pool.

Protocol such as Stratum (STM) or GetBlockTemplate (GBT)

Pool miners connect to the pool serve

Each pool miner then mines using the block template, at a higher (easier) target than the bitcoin network target, and sends any successful results back to the pool server to shares.

Peer-to-peer mining pool (P2Pool)

P2Pool works by decentralizing the functions of the pool server, implementing a parallel blockchain-like system called a share chain.

Rate of one share block every 30 seconds.

Shares forward from the previous share block. When one of the share blocks also achieves the bitcoin network target, it is propagated and included on the that preceded the winning share block.

Dedicated computer with enough disk space, memory, and internet bandwidth to support a full bitcoin node and the P2Pool node software. P2Pool miners connect their mining hardware to their local P2Pool node, which simulates the ers connect their mining naraware to their local PZPOOL

## Consensus Attacks

A consensus attack cannot steal bitcoin, spend bitcoin without signatures, redirect bitcoin, or otherwise change past transactions or ownership records.

Can only affect the most recent blocks and cause denial-of-service disruptions on the creation of future blocks.

In addition to a double-spend attack, the other scenario for a consensus attack is to deny service to specific bitcoin participants (specific bitcoin addresses). An attacker with a majority of the mining power can simply ignore specific transactions.

Despite its name, the 51% attack scenario doesn’t actually require 51% of the hashing Despi

Can be attempted with a smaller percentage of the hash

51% threshold is simply the level at which such an attack is almost ing power. The 51% thres.

Essentially a tug-of-war for the next block and the “stronger” group is more likely to win.

Possible with as little as 30% of the hashing power.

A malicious attack aimed at crippling bitcoin would quire enormous investment and covert planning, but could conceivably be launched by a well-funded, most likely state-sponsored, attacker.

## Changing the Consensus Rules

In order to evolve and develop the bitcoin system, the rules have to change from time to time to accommodate new features, improvements, or bug fixes. Unlike traditional software development, however, upgrades to a consensus system are much more difficult and require coordination between all the participants.

### Hard Forks

After the fork the network does not reconverge onto a single chain. In stead, the two chains evolve independently.

Different set of consensus rules than the rest of the network.

Because of a bug or because of a deliberate change in the implementation of the consensus rules.

Scam can be used to change the rules of consensus, but they require coordination between all participants in the system

Nodes that do not upgrade to the new | consensus rules are unable to participate in the consensus mechanism and are forced onto a separate chain at the moment of the hard fork.

Can be thought of as not “forward compatible,” in that nonupgraded systems can no longer process the new consensus rules.

Miners using the old rules may not even receive block 7b if all the nodes they are connected to are also obeying the old rules and therefore not propagating the block.

### Hard Forks: Software, Network, Mining, and Chain

In the case of a deliberate change to the consensus rules, a software fork precedes the hard fork.

There are numerous alternaimplementations of Bitcoin Core, and even software forks, that do not change the consensus rules and barring a bug, can coexist on the network and interoperate without causing a hard fork.

Developing in four stages: a software a network fork, a mining fork, and a chain fork.

Nodes following the original consensus rules will temporarily ban and disconnect from any nodes that are sending them these invalid transactions blocks

### Diverging Miners and Difficulty

### Contentious Hard Forks Contentious

Some developers are opposed to any form of hard fork, seeing it as too risky. Others see the mechanism of hard fork as an essential tool for upgrading the consensus rules in a way that avoids “technical debt” and provides a clean break with the past.

Finally, some developers see hard forks as a mechanism should be used rarely, with a lot of advanced planning an only under near unanimous consensus.

### Soft Forks

Change is implemented in such a way that unmodified client still sees the transaction or block as valid under the previous rules, the change can happen without a fork.

In practice, a soft fork is not a fork at all. A

Allows unupgraded clients to continue to operate change to the consensus rules that

Grades can only be used to constrain the consensus rules, not to expand them.

Soft forks redefining NOP opcodes

Bitcoin Script had ten opcodes reserved for future use, NOP1through NOP 10.

Interpreted as a null-potent operator, meaning they have no effect. Execution continues after the NOP opcode as if it wasn’t there.

Other ways to soft fork upgrade

Mechanism used for this is a modification of the locking script of UTXO created segwit rules, such that unmodified clients see the locking script as redeemable with any unlocking script whatsoever. As

### Criticisms of Soft Forks

Explicit goal of allowing nondisruptive upgrades.

Unacceptable tradeoffs. Common criticisms of soft fork changes include:

**Technical debt**

* Future cost of code maintenance because of design tradeoffs made in the past. Code complexity in turn increases the likelihood of bugs and security vulnerabilities.

**Validation relaxation**

* Unmodified clients are not validating using the full of consensus rules, as they are blind to the new rules.

**Irreversible upgrades**

* Create transactions with additional consensus constraints, they come irreversible upgrades in practice.
* If a soft fork upgrade were to be eversed after beings activated, any transactions created under the new rules result in a loss of funds under the old rules.

## Soft Fork Signaling with Block Version

Coordinate their actions, there is a signaling mechanism that allows them to show their support for a consensus rule change.

### BIP-34 Signaling and Activation

BIP-34, used the block version field to allow miners to signal readiness for a specific consensus rule change.

Prior to BIP-34, the block version was set to “1” by convention not enforced by consensus.

BIP-34 defined a consensus rule change that required the coinbase field (input) of the coinbase transaction to contain the block height and version greater than 2

BIP-34 defined a two-step activation mechanism, bas based on a rolling window of 1000 BIP-34

75% (750 of the most recent 1000 blocks) are marked with version “2” then version “2” blocks must contain block height in the coinbase transaction or they” | are rejected as invalid. Version “1” blocks are still accepted by the network and not need to contain block-height. The old and new consensus rules coexist do not need to

When 95% (950 of the most recent 1000 blocks) are version “2,” version “1” . blocks are no longer considered valid.

All blocks must comply with the new consensus rules, and all valid blocks must contain block-height in the coinbase transaction.

After successful signaling and activation under the BIP-34 rules, this mechanism was twice more to activate soft forks:

In or ’ Strict Encoding of Signatures was activated by BIP-34 style signaling with a block Encoding of Signatures was activated by BI-34

BIP-65 (<https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki>) CHECK LOCKTIMEVERIFY was activated by BIP-34 style signaling with a block version “4” and invalidating version “3” block

Retired and replaced with the BIP-9 signaling mechanism

### BIP-9 Signaling and Activation

Limitations:

By using the integer value of the block version, only one soft fork could be activated at a time, so it required coordination between soft fork proposals and agreement on their prioritization and sequencing.

Didn’t straight forward way to reject a change and then propose a different one.

Irrevocably reduced the available block versions for future changes.

Interprets the block version as a bit field instead of an integer. Leaves 29 bits that can be used to independently and simultaneously signal readiness on 29 different proposals.

Sets a maximum time for signaling and activation.

Proposal is not activated within the TIMEOUT period defined in the proposal), the proposal is considered rejected.

A data structure that contains the following fields:

Name distinguish between re

Bit through 28, the bit in the block version that miners use to signal approval for this proposal.

Starttime

The time (based on Median Time Past, or MTP) that signaling starts after which

Endime

The time (based on MTP) after which the change is considered rejected if it has not reached the activation threshold.

For every retarget period, if the sum of blocks signaling for a proposal exceeds 95% (1916 of 2016), the proposal will be activated retarget period later.

BIP-9 was first implemented for the activation of CHECKSEQUENCEVERIFY and associated BIPs (68, 112, 113). The proposal named “csv” was activated successfully in July of 2016.

## Consensus Software Development

‘Authority” that can impose its will on the participants of the network. Power is diffused between multiple constituencies such as miners, core developers, wallet developers, exchanges, merchants, and end users. Decisions cannot be made unilaterally any of these constituencies.

Participants must coordinate, or no changes can be made.

Perfect solution for consensus development. Both hard forks and soft forks involve tradeoffs.

Change is difficult and consensus forces compromise.

The system’s greatest strength.

# Bitcoin Security

## Security Principles

A centralized model, suc depends on access control and vetting to keep bad actors out of the system. BY comarison, a decentralized system like bitcoin pushes the responsibility and control to the users.

Because security of the network is based on Proof-of-Work, not access conol, the network can be open and no encryption is required for bitcoin traffic.

Contains the user’s private identifier (the credit card number).

The payment network has to be secured end-to-end with encryption and must ensure that no eavesdroppers or intermediaries can compromise the payment traffic,

A bitcoin transaction authorizes only a specific value to a specific recipient and cannot be forged or modified.

### Developing Bitcoin Systems Securely

### Root of Trust

Concept called the root of trust, which is a trusted core used as the foundation for the security of the overall system or application.

Security architecture is developed around the root of trust as a series of concentric circles, like layers in an onion, extending trust outward from the center.

A correctly validated blockchain uses the genesis block as the root of trust, building a chain of trust up to the current block .

## User Security Best Practices

### Physical Bitcoin Storage

### Hardware Wallets

### Balancing Risk

In their effort to prevent theft, the owners had implemented a complex series of encrypted backups. In

### Diversifying Risk

### Multisig and Governance

Can also offer redundancy, where a single person holds several keys that are stored in different locations.

### Survivability

Important security consideration that is often overlooked is availability, especially in the context of incapacity or death of the key holder.

# Conclusion

A completely new, unprecedented, and complex technology,

Better security tools and practices that are easier to use by nonexperts.

# Blockchain Applications

The features offered by the bitcoin blockchain, as an application platform. We will consider the application building primitives, which the building blocks of any blockchain application

## Introduction

Higher-level concepts of accounts, balances, and payments can be derived from these basic primitives, so can many other complex applications. Thus, the bitcoin blockchain can become an application platform offering trust services

## Building Blocks (Primitives)

* No Double-Spend
* Immutability is underwritten by energy
  + Increases with the amount of work committed on top of the block containing a transaction.
* Neutrality
  + Network propagates valid transactions regardless of the origin or content of those transactions.
* Secure Timestamping
  + Consensus rules reject any block whose timestamp is too far in the past or consensus rules reject any block whose timestamp is too
    - Unspent-before guarantee
* Authorization
  + Scripts that contain a requirement for a digital signature cannot be executed without authorization by the holder of the private key implied in the script.
* Auditability
  + Public and can be audited.
  + Unbroken chain to the genesis block.
* Accounting
  + The outputs cannot exceed the inputs.
* Nonexpiration
* Integrity
  + Nother SIGHASH type cannot be modified without invalidating the signature, thus invalidating the transaction itself.
* Transaction Atomicity
  + Either valid and confirmed (mined) or Bitcoin
  + There is no interim state for a transaction. At any point in time a transaction is either mined, or not.
* Discrete (Indivisible) Units of Value
  + Outputs are discrete and indivisible units of value.
  + Spent or unspent, in full.
* Quorum of Control
  + Scripts impose a quorum of authorization,
  + The M-of-N requirement is enforced bY the consensus rules.
* Timelock/Aging
* Replication
  + Durable and resilient to power loss, data loss, etc.
* Forgery Protection
  + It is not possible to create or counterfeit value.
* Consistency
  + Reorg or disagreement has exponentially decreasing likelihood, based on the depth at which they are recorded.
* Recording External State
  + Transaction can commit a data value, via OP\_RETURN, representing a state transition in an external state machine.
* Predictable Issuance

## Applications from Building Blocks Applications

* Proof-of-Existence (Digital Notary)
  + Immutability + Timestamp + Durability.
* Kickstarter (Lighthouse)
  + Consistency + Atomicity + Integrity.
  + Cannot be spent (Atomicity) until the goal (output value) is funded (Consiscannot
* Payment Channels
  + Quorum of Control + Timelock + No Double Spend + Nonexpiration + Censorship Resistance + Authorization

## Colored Coins

Use bitcoin transactions to record the creation, ownership, and transfer of extrinsic assets other than bitcoin.

* Not stored directly on the bitcoin blockchain, as
* Certificates of ownership
* Marking a nominal amount of bitcoin, for example, a single satoshi, to represent something other than the bitcoin value itself.
* Enhanced Padded-Order-Based Coloring or EPOBC, assigned extrinsic assets to a 1-satoshi output.
* Open Assets <http://www.openassets.org/>) and Colored Coins by Colu (<http://coloredcoins.org>). |

### Using Colored Coins

Use special colored-coin addresses to mitigate this risk and to ensure that colcoins are not sent to unaware wallets.

Must use a colored-coins explorer to interpret the metadata

### Issuing Colored Coins

Initial transaction, the issuance transaction registers asset on the bitcoin blockchain and creates an asset ID that is used to reference asset.

Can be divisible or indivisible, meaning that the amount of asset in a transfer can be an integer (e.g., 5) or have decimal subdivision (e.g., 4.321). As

Fixed Issuance, meaning a certain amount are issued only once, or can be reissued

Some colored coins enable dividends

### Colored Coins Transactions

The output containing the OpP\_RETURN is called the marker output.

T assigns specific values and colors to the other outputs by referencing their order in the transaction.

Marker output encodes an opcode determines how the metadata is interpreted

Opcodes OXU1 unrough OxOF indic

Metadata contain simple scripts that transfer specific amounts of assets from inputs to outputs, by reference to their index. Ordering of inputs and outputs is therefore important in the interpretation of the script.

Other “tricks” to store metadata in a transaction. Examples include putting metadata a redeem script, followed by OP\_DROP opcodes to ensure the script ignores the metadata -

Another mechanism used is a 1-of-N multisig script

If the metadata is too long to fit in OP\_RETURN, the colored coins protocol may use

## Counterparty

A decentralized exchange for assets. Counterparty is implementing smart contracts, based on the Ethereum Virtual Machine (EVM).

Content example, Tokenly is platform built on top of Counterparty LITUS W be used to rent, access, trade, or shop for content, products, and services.

## Payment Channels and State Channels

Between two parties, outside of the bitcoin blockchain.

Promissory notes for eventual batch settlement.

High transaction throughput, low (submillisecond) latency, and fine (satoshi-level) granularity.

Constructs represented by the exchange of state between two parties, outside of the blockchain.

One-way (unidirectional) payment channel for a metered micropayment service, such as streaming video.

### State Channels-Basic Concepts and Terminology

A transaction that lock: nared state on the blockchain. This is called the funding transaction or anchor transshare

Single transaction must be transmitted to the network and mined to This single tran

Two parties then exchange signed transactions, called commitment transactions,

Invalidate the previous states, so that the most up-to-date commitment transaction is always the only that can be redeemed

Channel can be closed either cooperatively, by submitting a final settleFinally, the channel can be closed either cooperativ bY either party submitting the last commitment transaction to the blockchain. A

Final state of the channel and is settled on the blockchain.

Parties can exchange any number of commitment transactions are never seen by anyone else, nor submitted to the blockchain.

### Simple Payment Channel Example

Setting the channel capacity

### Making Trusted Channels

Problems can be solved with transaction-level timelocks (nLocktime).

Problem, Emma constructs the funding and refund transaction transmits only the refund transaction to Fabian and obtains his signature.

Each commitment transaction setting a shorter timelock, allowing it to be spent before the previous commitments become valid.

Maximum timelock when the channel is first opened, they limit the lifetime of the channel. Worse, they force channel implementations to strike a balance between allowing long-lived channels and forcing one of the participants to wait a very long for a refund case of premature closure.

Each subsequent commitment transaction must decrement the timelock, there is an explicit limit on the number of commitment transactions that can be exchanged between the parties.

make that balance a reality, they can construct a settlement transaction without a timelock representing that same balance. In a cooperative close, either party takes the identical in every way except that it omits the timlock.

## Asymmetric Revocable Commitments

Constraints that make payment channels difficult to use.

Even though a transaction cannot be canceled, it can be constructed in such a way as make it undesirable to use. The way we do that is by giving each party a revocation key that can be used to punish the other party if they try to cheat.

First proposed as part of the Lightning Network

A commitment transaction with two outputs. The first output pays Irene bitcoin she is owed immediately. The second output pays Hitesh the 5 bitcoin he is owed, but only after a timelock of 1000 blocks.

CHECKSEQUENCEVERIFY

each party has a commitment transaction, spending the 2-of-2 funding output. This input is signed by the other party.

At any time the party holding the traction can also sign (completing the 2-of-2) and broadcast.

Has to revoke this ment transaction before Irene agrees to sign the next commitment transaction. To

In each round, as the channel state revocation protocol is bilateral, meaning that in ea L וי the previous commitments,

Crucially. Before signing new commitment transactions, they must first exchange revocation keys to invalidate the prior commitment.

Relative time locks (CSV) are a much better way to implement payment channels and a very significant innovation in this technology. With this construct, the channel can remain open indefinitely and can have -billions of intermediate commitment transactions.

Billions of intermediate commitment transactions. In prototype implementations of more than 281 trillion (2.8 x 1014) state transitions in any single channel!

### Hash Time Lock Contracts (HTLC)

To create an HTLC, the intended ecipient of the payment will first create a secret R. They then calculate the hash of recipient of the

Included in an outputs locking script. Whoever knows the secret can use it to redeem the output. The secret R is also referred to as a preimage to the hash function.

Data that is used as input to a preimage to the

If the secret is not revealed, the payer of the HTLC can get a “refund” after some time. This is achieved with an absolute time lock using CHECKLOCKTIMEVERIFY.

By exercising the first clause of the IF flow.

After a certain number of blocks payer can claim a refund using the second clause in the IF flow.

Adding a CHECKSIG operator and a public key in the first clause restricts redemption of the hash to a named recipient, who must also know the secret R.

## Routed Payment Channels (Lightning Network)

### Basic Lightning Network Example

Through a series of HTLC commitments on the payment channels connecting the participants.

The extra 0.003 will be used to compensate the intermediate nodes for their participation in this payment route.

With this commitment in hand, Bob’s constructs an HTLC on his payment channel with Carol. B

Bob knows that if Carol can claim his HTLC, she has to produce u P Figure 12-10 step 3). Bob knows that avı R. If Bob has R in nine blocks, he can use it to claim Alice’s HTLC to him. H

At this hop in the route, Eric has secret R. He can therefore claim the HTLC bY, Diana

### Lightning Network Transport and Routing

Communications between LN nodes are encrypted point-to-point. In addition, nodes have a long-term public key that they use as an identifier and to authenticate other

First construct a Whenever a node wishes to send a payment to another out

Connecting her node to Eric’s node. Once Alice’s node nas series of encrypted and nested instructions to connect each of the adjacent payment series of

Path is only known to Alice’s node.

Implements an onion-routed protocol based on a scheme called Sphinx

Verify and decrypt their portion of route information

Other than the previous and next hops, they cannot learn about any other nodes that are part of the path.

They cannot identify the length of the payment path, or their own position in canno

Path is encrypted in such a way that a network-level attacker cannot associate the packets from different parts of the path to each other.

Unlike Tor (an onion-routed anonymization protocol on the internet), there are no “exit nodes” that can be placed under surveillance.

How it is possible that the nodes do not know the length of the path and their position in that path.

The path is always fixed at 20 hops and padded with random data. Each node sees the next hop and a fixed-length encrypted message to forward. Only the final recipient sees that there is no next hop. To everyelse it seems as if there are always 20 more hops to go.

### Lightning Network Benefits

Gain a significant increase in capacity, privacy, granularity, and speed, without sacrificing the principles of trustless operation without intermediaries:

* Privacy
  + Much more private
  + They do not know the “sender or recipient
* Fungibility
  + Much more difficult to apply surveillance and blacklists on bitcoin, inincreasing the fungibility of the currency.
* Speed
  + settled in milliseconds, rather minutes, as HTLCs are cleared without committing transactions to a block.
* Granularity
  + enable payments at least as small as the bitcoin “dust” limit, perhaps even smaller. Some proposals allow for sub satoshi increments.
* Capacity
  + orders of magnitude. There is no practical upper bound
* Trustless Operation

## Conclusion

Emerging applications expand the scope of bitcoin beyond payments and beyond financial instruments, to encompass many other applications where trust is critical.

By decentralizing the basis of trust, the bitcoin blockchain is a platform that will spawn many revolutionary applications in a wide variety of industries.

# The Bitcoin Whitepaper by Satoshi Nakamoto

# Transaction Script Language Operators, Constants, and Symbols

* + Table B-1. Push value onto stack
  + Table B-2. Conditional flow control
  + Table B-4. Stack operations
  + Table B-5. String splice operations
  + Table B-6. Binary arithmetic and conditionals
  + Table B-7. Numeric operators
  + Table B-8. Cryptographic and hashing operations
  + Table B-9. Nonoperators
  + Table B-10. Reserved OP codes for internal use by the parser

# Bitcoin Improvement Proposals.

* + Design documents providing information to the community, or for describing a new feature for bitcoin or its processes or environment.
  + BIP-01 BIP Purpose and Guidelines, there are three kinds of BIPS:
    1. Standard BIP
       1. A change in block or transaction validity rules, or any change or addition that affects the interoperability of applications using bitcoin.
    2. Informational BIP
       1. Provides general guidelines or information to the bitcoin community, but does not propose a new feature
       2. Do not necessarily represent a bitcoin community consensus or recommendation, so users and implementors may ignore informational BIPs or follow their advice.
    3. Process BIP
       1. Process BIPs are like standard BIPs but apply to areas other than the bitcoin protocol itself
  + Outline of BIP-1 through BIP-199

# Bitcore

* + Suite of tools provided by BitPay.
  + Provided by BitPay. Its goal is to provide easy-to-use toois f some modules written specifically for NodeJS. Finally, the “node” module of Bitcore includes Bitcoin Core’s C++ code. Please se

# pycoin, ku, and tx

* + Python library pycoin C. supports manipulation of bitcoin keys and transactions, even supporting the scripting language enough to properly deal with nonstandard transactions.
  + Key Utility (KU)
    1. Command-line utility ku (“key utility”) is a Swiss Army knife for manipulating It
  + Transaction Utility (TX)
    1. Display transactions in human-readable form, fetch Transactions from pycoin’s transaction cache or from web services (block.

# Bitcoin Explorer (bx) Commands

* + A command-line tool that offers a variety of commands for management and transaction construction.
  + Examples of bx Command Use
  + Seed can be encoded using the mnemonic-encode command:

# Index.

* + List of key terms and page numbers for reference

### 