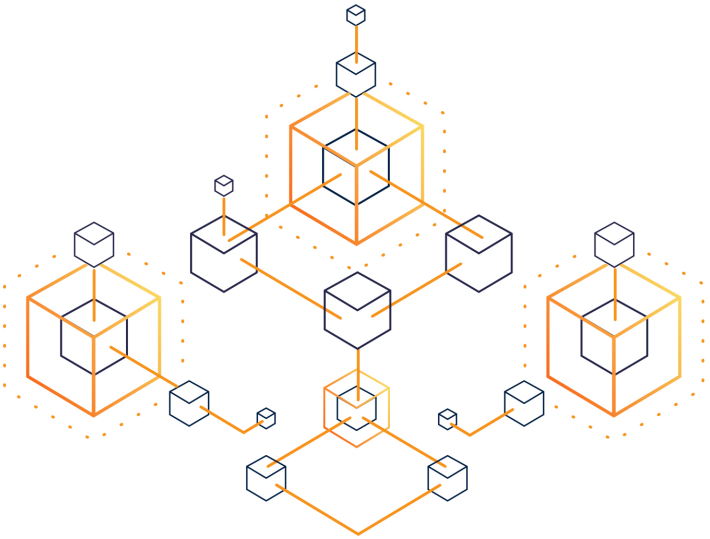
INVENTING BITCOIN



THE TECHNOLOGY BEHIND THE FIRST TRULY SCARCE AND DECENTRALIZED MONEY EXPLAINED

YAN PRITZKER

Get the updated version online at inventingbitcoin.com

This book is dedicated to my parents Yury and Lana, who extracted our family from the former USSR, an autocratic socialist regime with tight capital controls.

It's also dedicated to my wife Jessica, who has had to endure my inability to stop talking about Bitcoin, and my staying up late to finish this book.

INTRODUCTION

Welcome to *Inventing Bitcoin*! My goal with this book isn't to analyze the economics of Bitcoin or convince you that Bitcoin is the new digital gold; for that, I'll point you to *The Bitcoin Standard* by Saifedean Ammous.

I'm not going to look at Bitcoin from the standpoint of investing, or try to convince you that everyone should own a little. We're not going to look at charts or price histories. For that, I'll recommend *Cryptoassets* by Chris Burniske and Jack Tatar.

Finally, we're not going to dig into how the protocol works on a deep level, and we're not going to look at any computer code. For that, I recommend the seminal *Mastering Bitcoin* by Andreas Antonopulous.

My goal is simply to tickle your brain and give you a taste of the computer science and economic game theory that make Bitcoin one of the most interesting and profound inventions of our time.

Most people, upon first hearing about Bitcoin, don't really understand it. Is it magical Internet money? Where does it come from? Who controls it? Why is it important?

For me, understanding all the things that come together to make Bitcoin work—the physics, math, cryptography, game theory, economics, and computer science—was a profound moment. In this book, I hope to share this knowledge with you in a very simple and easy to understand way.

The way we'll do it is one step at a time. With nothing but a high school level math background, we will walk through *inventing bitcoin*, step by step. I hope that this book will give you just enough of a taste to send you down the Bitcoin rabbithole. Let's get started!

1

WHAT IS BITCOIN?

Bitcoin is a *peer to peer electronic cash*, a digital money that can be transferred between people or computers without any trusted middleman (such as a bank), and whose issuance is not under the control of any single party.

Think of a paper dollar or physical metal coin. When you give that money to another person, they don't need to know who you are. They just need to trust that the cash they get from you is not a forgery. Typically people do this with physical money using just their eyes and fingers.

As we've shifted to a digital society, the majority of our payments are now made digitally over the Internet by means of a middleman service: a credit card company like Visa, a digital payment provider such as PayPal or Apple Pay, or an online platform like WeChat in China.

The shift to digital payments brings with it the reliance on a central actor that has to approve and verify every payment. This is because the nature of money has changed from a physical thing you can look at and touch and verify yourself, to digital bits that have to be verified by the party that controls their transfer.

Bitcoin offers an alternative to centrally controlled digital money with a system of three basic components. We'll get into the motivations behind this design in the next section.

1. A digital asset (typically *bitcoin* with a lowercase *b)* with a supply that is limited, known in advance, and unchangeable. This stands in stark contrast to the money most of us are used to today, which are notes issued by governments or central banks whose supply expands at an unpredictable rate over time.
2. A bunch of interconnected computers (the *Bitcoin network)*, which anyone can join. This network serves to track ownership of bitcoins and to transfer them between participants, bypassing any middlemen such as banks, payment companies, and government entities.
3. The Bitcoin client software, which is a piece of code that anyone can run on their computer to become a participant in the network. This software is open source, which means that anyone can see how it works, as well as contribute new features and bug fixes to it.

Where Did It Come From?

Bitcoin was invented by a person or group known by the pseudonym of Satoshi Nakamoto around 2008. No one knows the identity of this

person or group, and as far as we know, they've disappeared and haven't been heard from for years.

On Feb 11, 2009, Satoshi revealed the first prototype of Bitcoin on an online forum for cypherpunks, people who work on cryptography technology and are concerned with individual privacy.

I've extracted the relevant bits below. In the next section, we'll explain some of these statements and Satoshi's motivations for the invention of Bitcoin.

> *I've developed a new open source P2P e-cash system called Bitcoin. It's completely decentralized, with no central server or trusted parties, because everything is based on crypto proof instead of trust. [...]*
>
> *The root problem with conventional currency is all the trust that's required to make it work. The central bank must be trusted not to debase the currency, but the history of fiat currencies is full of breaches of that trust. Banks must be trusted to hold our money and transfer it electronically, but they lend it out in waves of credit bubbles with barely a fraction in reserve. We have to trust them with our privacy, trust them not to let identity thieves drain our accounts. Their massive overhead costs make micropayments impossible.*
>
> *A generation ago, multi-user time-sharing computer systems had a similar problem. Before strong encryption, users had to rely on password protection to secure their files [...]*
>
> *Then strong encryption became available to the masses, and trust was no longer required. Data could be secured in a way that was physically impossible for others to access, no matter for what reason, no matter how good the excuse, no matter what.*
>
> *It's time we had the same thing for money. With e-currency based on cryptographic proof, without the need to trust a third party middleman, money can be secure and transactions effortless. [...]*

> *Bitcoin's solution is to use a peer-to-peer network to check for double-spending. In a nutshell, the network works like a distributed timestamp server, stamping the first transaction to spend a coin. It takes advantage of the nature of information being easy to spread but hard to stifle. For details on how it works, see the design paper at http://www.bitcoin.org/bitcoin.pdf*
>
> — Satoshi Nakamoto

When Bitcoin was launched, only a handful of people used it and ran computers (called *nodes*) to power the Bitcoin network. Most people at the time thought it was a joke, or that the system would reveal serious design flaws that would make it unworkable.

Over time, more people joined the network, using their computers to add security to the network and reinforcing that it had value by exchanging other currencies for it, or accepting it for goods and services. Today, ten years later, it is used by millions of people with tens to hundreds of thousands of nodes running the free Bitcoin software, which is developed by hundreds of volunteers and companies worldwide.

Bitcoin was not an invention made in a vacuum. In his paper, Satoshi cited several important attempts at implementing similar systems including Wei Dai's b-money, and Adam Back's Hashcash. The invention of Bitcoin stood on the shoulders of giants, and yet it was profound in its simplicity in creating the first truly decentralized—that is, not under any one person's control—system for issuing and transferring a digital money.

What Problems Does it Solve?

Let's break down some of Satoshi's post. Throughout the book, we will cover how these concepts are actually implemented. It's not important

that you immediately understand all the unfamiliar concepts in this section, but you'll want to see what Satoshi's goals were, so that we can aim to achieve them as we go through the exercise of *Inventing Bitcoin.*

I've developed a new open source P2P e-cash system

P2P stands for *peer to peer* and indicates a system where one person can interact with another without anyone in the middle, as equal peers. You may recall P2P file sharing technologies like Napster, Kazaa, and BitTorrent, which first enabled people to share music without downloading it from a website. Satoshi designed Bitcoin to allow people to exchange *e-cash*, electronic cash, without going through a middleman in much the same way.

The software is *open source*, which means that anyone can see how it works and contribute to it. This is important as it removes the requirement of trust in Satoshi. We don't need to believe anything Satoshi wrote in his post about how the software works. We can look at the code and verify how it works for ourselves. Even better, if we don't like something, we can change it.

It's completely decentralized, with no central server or trusted parties...

Satoshi mentions that the system is *decentralized* to distinguish it from systems that do have central control. Prior attempts to create digital cash such as DigiCash by David Chaum in 1989 were backed by a *central server*, a computer or set of computers that was responsible for issuance and payment verification, run by one company.

Such centrally controlled private money issuance schemes were doomed to failure; people can't rely on a money that can go away when the company goes out of business, gets hacked, suffers a server crash, or is shut down by the government.

The decentralized nature of Bitcoin brings back the concept of cash to the digital realm: you can transfer it without talking to anyone, without

asking for permission, 24 hours a day, 365 days a year without going through any trusted party.

...everything is based on crypto proof instead of trust

How does Bitcoin get rid of the requirement of *trust*? We'll dive into this later in the book, but the basic idea is that instead of trusting someone that says "I am Alice" or "I have $10 in my bank account" on their word, we can use cryptographic math to state the same facts in a way that is very easy to verify by the recipient of the proof. This would become the basis of Bitcoin's system for proving ownership of funds as well as providing security for the network.

We have to trust [the banks] with our privacy, trust them not to let identity thieves drain our accounts

Unlike using your bank account, digital payment system, or credit card, Bitcoin allows two parties to transact without giving up any personally identifying information.

Centralized repositories of consumer data stored at banks, credit card companies, payment processors, and governments are giant honeypots for hackers. As if to prove Satoshi's point, Equifax was massively compromised in 2017, leaking the identities and financial data of more than 140 million people to hackers.

Bitcoin aims to decouple financial transactions from real world identities. After all, when we give physical cash to someone, they don't need to know who they are, nor do we need to worry that after our exchange they can use some information we gave them to steal more of our money. Why shouldn't we expect the same, or better, from digital money?

The central bank must be trusted not to debase the currency, but the history of fiat currencies is full of breaches of that trust

Fiat, which is Latin for "let it be done," refers to government and central-bank issued currency which is decreed as legal tender by the

government. Historically, money was freely chosen by market participants from things that were hard to produce, easy to verify, and easy to transport, such as salt, seashells, stones, silver, and gold.

We slowly shifted from a world economy that used gold as money to one where paper came to represent a claim on that gold. Eventually, the paper was entirely separated from any physical backing by Nixon, who ended the international convertibility of the US dollar to gold in 1971. The end of the gold standard allowed governments and central banks to increase the money supply at will, diluting the value of each note in circulation, known as *debasement*. Although government-backed, redeemable for nothing, pure fiat currency is the money we all know and use day to day, it is actually a relatively new concept, only about a century old.

We trust our governments not to abuse their printing press, but we don't need to look far in history for examples of *breaches of that trust*. In autocratic and centrally planned socialist regimes where the government has their finger directly on the money machine, such as Venezuela, the currency has become nearly worthless. The Venezuelan Bolivar went from 2 Bolivar to the dollar in 2009 to 250,000 Bolivar to the dollar in 2019. As I write this book, Venezuela is in the process of collapse and regime change due to the terrible mismanagement of capital by its government.

In contrast to the centrally issued *fiat* currency whose supply cannot be predicted, in order to prevent *debasement*, Satoshi designed a system of money where the supply was fixed and issued at a predictable and unchangeable rate. There will only ever be 21 million bitcoins, though each bitcoin can be divided into 100 million units now called satoshis.

Prior to Bitcoin, digital assets were not scarce. It is easy to copy a digital book, audio file, or video and send it to your friend. The only exceptions to this are digital assets controlled by middlemen. For example, when you rent a movie from iTunes, you can watch it on your device only because iTunes controls the delivery of the movie and can stop it

after your rental period is up. Similarly, your digital money is controlled by your bank. It is the bank's job to keep a record of how much money you have, and if you transfer it to someone else, they can authorize or deny such a transfer.

Bitcoin is the first digital network which enforces scarcity without any middlemen and is the first asset known to humanity whose unchangeable supply and schedule of issuance is known completely in advance. Not even precious metals like gold have this property, since we can always mine more and more gold deposits at an unpredictable rate if it is profitable to do so. We'll get into how this is enforced later.

Data could be secured in a way that was physically impossible for others to access [...] It's time we had the same thing for money

Our current systems of securing money, such as putting it in a bank, rely on trusting someone else to do the job. Trusting such a middleman not only requires trust that they won't do something malicious or foolish, and that hackers won't steal it, but also that the government won't seize or freeze your funds. Yet it has been demonstrated across the world, time and time again, that governments can and do shut down access to money when they feel threatened.

It might sound silly to someone living in the United States, or another highly regulated economy, to contemplate waking up with your money gone. As an anecdote, I've had my funds frozen by PayPal simply because I hadn't used my account in months. It took me over a week to get restored access to "my" money. I'm lucky to live in the United States, one of few countries where at least I could hope to seek some legal relief if PayPal froze my funds, and where I have basic trust that my government and bank won't steal my money.

Much worse things have happened, and are currently happening, in countries with less freedom, such as banks shutting down during currency collapses in Greece, banks in Cyprus using bail-ins by stealing funds from their customers, or the government declaring certain bank

notes worthless in India, stripping people of their wealth, causing runs on ATM machines and people starving and dying due inability to access their capital.

The former USSR, where I grew up, had a tightly centrally controlled economy leading to massive shortages of goods. When we wanted to leave, we could only exchange a limited amount of money per person under an official government controlled exchange rate that was vastly divorced from the true free market rate.

When economies start failing in autocratic countries, they tend to implement strict economic controls, preventing people from withdrawing their money from banks, carrying it out of the country, or exchanging it for not-yet-worthless currencies like the US dollar on the free market.

Bitcoin provides a system of security that does not rely on trust in a third party to secure your money. Instead Bitcoin makes your coins *impossible for others to access* without a special key that only you hold, *no matter for what reason, no matter how good the excuse, no matter what*.

Bitcoin separates money and state, and thus provides a check on autocrats and dictators, restoring freedom of control over wealth, and the freedom to transport your wealth across borders without interference.

Bitcoin's solution is to use a peer-to-peer network to check for double-spending [...] like a distributed timestamp server, stamping the first transaction to spend a coin

A *network* simply refers to the idea that a bunch of computers are connected and can send messages to each other. The word *distributed* means that there is not a central party in control, but rather that all the participants coordinate to make the network successful.

In a system without central control, it's important to know that nobody is cheating. The idea of *double-spending* refers to the ability to spend the same money twice. Satoshi is describing that the participants of the

Bitcoin network work together to *timestamp* (put in order) transactions so that we know what came first, in order to prevent this digital equivalent to forging money. In the next chapters, we will build this system from the ground up. It will enable us to detect forgery without relying on any central issuer or transaction validator.

The invention of Bitcoin solved a number of interesting technical problems in service of solving the issues of privacy, debasement, and central control in current monetary systems:

1. How to create a peer to peer network that anyone can voluntarily join and participate in.
2. How a group of people that don't have to reveal their identities or trust each other can maintain a shared ledger of value, even if some of them are dishonest.
3. How to create true digital scarcity without a middleman.
4. How to create a digital asset that is unforgeable, instantly verifiable, and resistant to theft and hacking.

Let's figure out how we can build this system!

2

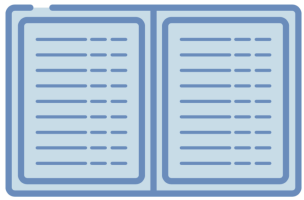
REMOVING THE MIDDLEMAN

In the prior chapter, we discussed that Bitcoin provides a peer to peer system for the transfer of value. Before we can dive into how that works, let's first understand how a traditional bank or payment company deals with tracking asset ownership and transfers.

Banks are Just Ledgers

How does a digital payment made by your bank, PayPal, or ApplePay work? Very simply, these middleman entities have a ledger of accounts and transfers.

In this example, we will say *bank* but we really mean any other party that processes payments. We start with a Ledger of accounts that shows that Alice and Bob deposited money with the bank.



Bank's Ledger

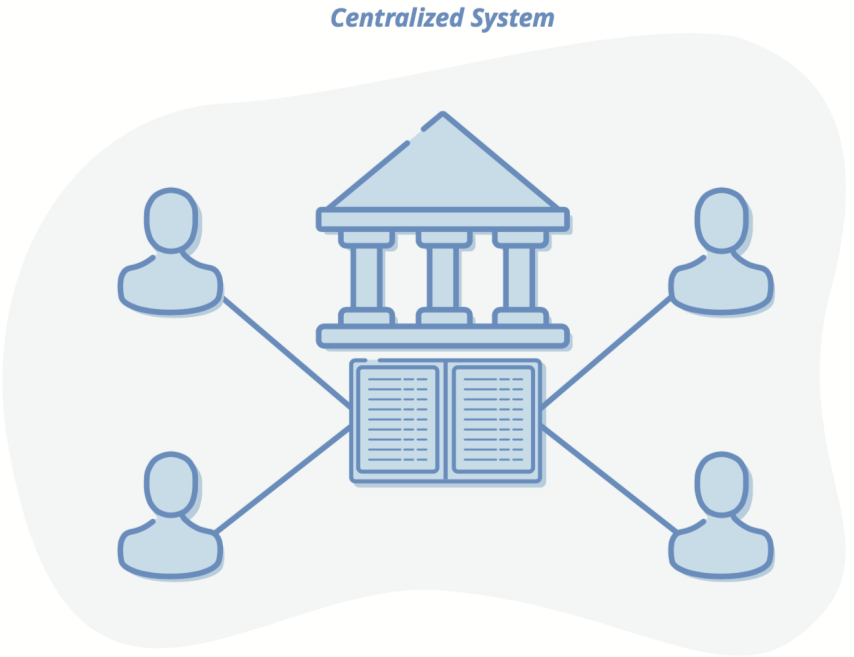
1. Alice: Credit for Cash Deposit +$2
2. Bob: Credit for Cash Deposit +$10

When Alice wants to send $2 to Bob, she calls her bank or uses a web or mobile wallet produced by her bank, authenticates herself to the bank using a username and password or pin code, and then puts in the request to transfer. The bank records it in their ledger.

Bank's Ledger

1. Alice: Credit for Cash Deposit +$2
2. Bob: Credit for Cash Deposit +$10
3. Alice: Debit -$2
4. Bob: Credit +$2

So the bank has recorded the new debits and credits, and now the money has moved. Simple!



The Double-Spending Problem

What happens if Alice now tries to spend those two dollars again? This is called the Double-Spending Problem. She files the request to the bank, but the bank says "Sorry, we see you've already spent $2 to pay Bob, you have no more money to send."

When you have a central authority like a bank, it is very easy for the bank to tell that you're trying to spend the cash you've already spent. That's because they're the only ones that get to modify the ledger, and they have internal processes including backup systems and audits by computers and humans to make sure it's correct and hasn't been tampered with.

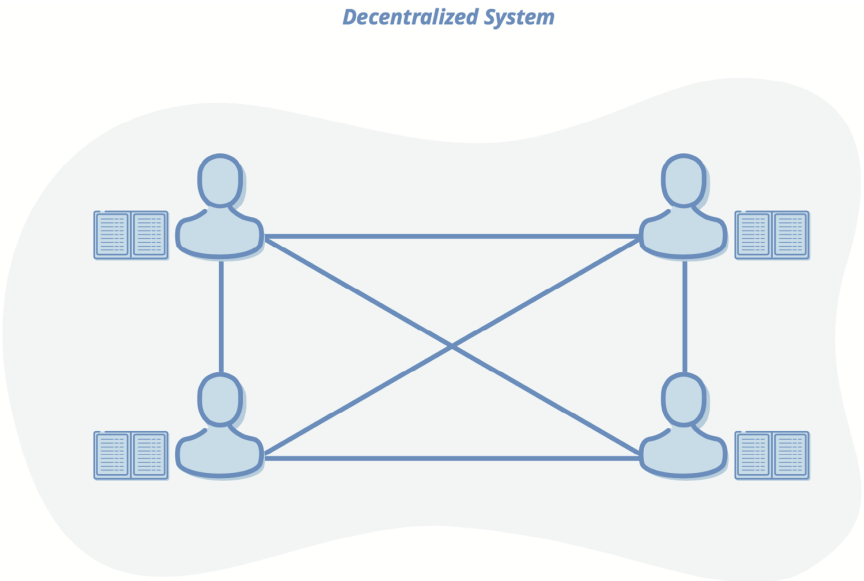
We call this a *centralized* system because it has a single point of control.

Let's Decentralize The Ledger

The first problem Bitcoin aims to solve is the removal of a trusted middleman by creating a *peer to peer* system. Let's imagine that banks have gone away and we need to recreate our financial system. But this time, we aren't going to create a central party. How can we maintain a ledger without any central party?

If we don't have one central ledger keeper, it must be the case that the ledger now belongs to the people. Vive la révolution. Here's how we do it.

First, a bunch of us get together and create a *network*. This just means we have some way to talk to each other. Let's say we exchange phone numbers or Snapchat accounts. When Alice wants to send money to Bob, instead of calling the bank, she Snapchats all her friends and tells them: "I'm sending $2 to Bob." Everyone acknowledges, replying "cool, we got it," and writes it into their own copy of the ledger. The picture now looks like this:



So now, instead of a single bank, we have a copy of the ledger in everyone's hands. Every time someone wants to spend money, they simply have to call or snapchat all their friends and let them know this is happening. Everyone records the transactions. Since the ledger is no longer in one place, we call it *distributed,* and because no central party is in charge, we call it *decentralized*.

How does this solve the double-spend problem? Well, since everyone has a copy of the ledger, if Alice tries to re-spend the $2 she already sent to Bob, her transaction would be rejected by everyone on the network, since they would consult their ledgers and tell her that according to their records, she already spent the money.

We now have a peer to peer network for recording ownership and transfers of funds. This system works very well between a group of friends that have social reasons not to cheat each other, but this doesn't scale to larger networks of people. As more and more people start to use the system, there is an incentive to cheat in order to give oneself extra money in the ledger.

How do we keep everyone honest?

3

TRUSTLESS AND PERMISSIONLESS

As long as our distributed ledger requires permission to join, and we can trust every party to be honest, the system works. But this kind of design can't scale to be used by millions of people worldwide.

Distributed systems made of arbitrary participants are inherently unreliable. Some people might occasionally go offline. That means they may not hear about our transactions when we broadcast them. Others may be actively trying to defraud us by saying that certain transactions happened or didn't happen. New people may join the network and get conflicting copies of the ledger. Let's take a look at how someone might try to cheat.

The Double-Spend Attack

If I'm Alice, I can *collude* with some of the other folks and tell them: "when I spend money, don't write it into your ledgers. Pretend like it never happened." Here's how Alice can perform a Double Spend Attack.

Starting with a balance of $2, Alice does the following:

1. She sends her $2 to Bob, to buy a candy bar. Now she should have $0 left.
2. David, Eve, and Farrah are colluding with Alice and do not write the transaction from Alice to Bob into their ledgers. In their copy, Alice never spent her money.
3. Charlotte is an honest ledger keeper. She notes the transaction from Alice to Bob. In her ledger, Alice has $0.
4. Henry was on vacation for a week and hasn't heard about any of these transactions. He joins the network and asks for a copy of the ledger.
5. Henry gets 4 false copies (David, Eve, Farrah, Alice) and one honest copy (Charlotte). How does he determine which one is real? With no better system, he goes for majority vote and is duped into accepting the fake ledger as the correct one.
6. Alice buys a candy bar from Henry using the $2 she doesn't actually have. Henry accepts it because for all he knows, Alice still has $2 in her account (according to the ledger he got from everyone else).
7. Alice now has 2 candy bars, and $4 of fake money has been created in the system. She pays off her friends in candy bars, and they repeat the attack 100 times on every new person who joins the network.
8. Alice is now holding all the candy bars and everyone else is holding large bags of fake money.
9. When they try to spend the money Alice supposedly sent them, David, Eve, and Farrah who control the majority of the network, reject these spends because they know the money is fake to begin with.

We have a problem. This is called a *consensus failure.* The people in the network did not come to consensus on what the state of reality is. Having no better system, they went with majority rule, which led to dishonest people controlling the network and spending money they didn't have.

If we want to make a *permissionless* system where anyone can participate without asking, then it must also be resilient to dishonest actors.

Solving the Distributed Consensus Problem

Now we get to solve one of the hardest problems in computer science: distributed consensus between parties where some are dishonest or unreliable. This problem is known as the Byzantine Generals Problem and is the key that Satoshi Nakamoto used to unlock the invention of Bitcoin. Let's start simple.

We need to get a bunch of people to agree on the entries in the ledger without knowing which ledger keepers have been writing down all the transactions correctly and honestly.

One naive solution is simply to appoint honest ledger keepers. Instead of everyone getting to write to the ledger, we pick a handful of friends like Charlotte, Gary, Frank, and Zoe to do it, because they don't tell lies and everyone knows they never party on the weekends.

So every time we have to do a transaction, instead of telling all our friends, we just call up Charlotte and the gang. They're happy to maintain the ledger for us for a small fee. After they write to the ledger, they call everyone else and tell them about the new ledger entries, which everyone still keeps as backup.

This system works really well, except one day, government agents show up and they want to know who's been running this shadow financial system. They arrest Charlotte and friends and take them away, putting an end to our distributed ledger. We all have unreliable backups, can't trust each other, and can't figure out whose backup should be used to start a new system.

Instead of a full shutdown, the government can also threaten our ledger keepers quietly with jail time if they accept transactions to Alice (who is suspected of selling drugs). The system is now effec-

tively under central control and we can't call it permissionless anymore.

What if we try democracy? Let's find a pool of 50 honest people and we'll run elections every day to keep rotating who gets to write to the ledger. Everyone in the network gets a vote.

This system works great until people show up and use violence or financial coercion to achieve the same ends as before:

1. Coerce the electorate to vote for the ledger keepers of their choosing.
2. Coerce the elected ledger keepers to write fake entries into the ledger.

We have a problem. Any time we appoint specific people to maintain the ledger, they must be trusted to be honest, and we have no way of defending them from being coerced by someone to do dishonest deeds and corrupting our ledger.

Mistaken Identity and Sybil Attacks

So far we looked at two failed methods of ensuring honesty: one used specific known ledger keepers, and the other used elected and rotating ledger keepers. The failure of both systems was that the basis of our trust was tied to real-world identity: we still had to specifically identify the individuals that would be responsible for our ledger.

Whenever we assume trust based on identity, we open ourselves up to something called a Sybil Attack. This is basically a fancy name for impersonation; it's named after a woman with multiple personality disorder.

Have you ever received a weird text from one of your friends only to find out her phone had been hijacked by her brother? When it comes to billions or even trillions of dollars at stake, people will justify all

kinds of violence in order to steal that phone and send that text. It is imperative that we prevent the people who get to keep our ledger for us from being coerced in any way. How do we do this?

Let's Build a Lottery

If we don't want the possibility of people being compromised by threats of violence or bribery, we need a system with so many participants that it would be impractical for anyone to coerce them. It must be the case that anyone at all can participate in our system, and that we don't have to introduce any kind of voting, which is subject to coercion by violence and vote buying.

What if we ran a lottery where we picked someone at random every time we wanted to write to the ledger? Here's our first design draft:

1. Anyone at all in the world can participate. Tens of thousands of people can join our ledger keeper lottery network.
2. When we want to send money, we announce to the entire network the transactions we want to perform, just like we did before.
3. Every ten minutes, we will select a winner (How? Not sure yet).
4. When we select a winner, that person gets to write all the transactions that they just heard about into the ledger.
5. If the person writes valid transactions (as deemed by all other network participants) into the ledger, they are paid a fee.
6. Everyone maintains a copy of the ledger, adding the information that the latest lottery winner produced.
7. We use the ten minute interval to make sure most people have time to update their ledgers between lottery runs.

This system is an improvement. It's impractical to compromise the participants of this system because it's impossible to know who the next winner will be. Still, this system has a few problems. What are they?

The Self-Run Lottery System

This lottery system as designed has two major problems:

1. Who will sell the tickets to the lottery and pick the winning numbers, if we have already determined that we can't have any kind of central party that can be compromised running it?
2. How do we ensure that the winner of the lottery actually writes good transactions into the ledger rather than trying to cheat the rest of us?

If we want a *permissionless* system that anyone can join, then we have to remove the requirement of trust from the system and make our system *trustless*. We have to come up with a system that has the following properties:

1. It must be possible for everyone to generate their own lottery ticket, since we can't trust a central authority.
2. It must be easy for all other participants to verify that you've won the lottery solely by examining your ticket, since we can't trust anyone to keep a record of the winning combination.
3. We must have some way to punish you if you win the lottery but write invalid transactions into the ledger, replacing trust in specific people with trust in incentives and punishments.

Let's tackle these one at a time. The full explanation of how this lottery works is probably the hardest thing to understand about Bitcoin, so we'll devote the next three chapters to covering the solution in depth.

Standard centralized lottery systems like Powerball are run by someone generating a random set of numbers and a bunch of tickets with random numbers on them. Only one ticket has the numbers exactly matching the secret random number known only to the organization

that runs Powerball. Since we can't rely on central authority, we must allow anyone to generate their own random numbers.

How will we verify the winner? In Powerball, the lottery owners know the winning combo. Since we can't have that in a decentralized system, we can instead create a system where everyone can agree on a number range ahead of time, and if your random number falls within the range, you win the lottery. We'll use a cryptographic trick called a hash function to do this. We'll dive into a light introduction to hashing in Chapter 4.

Finally, we must have a way to punish you if you cheat. Generating random numbers, i.e. lottery tickets, is basically free. How do we make it so that you actually have to spend money to buy tickets when there is no one you can buy them from? We'll make you buy them from the universe by spending energy, a scarce resource that cannot be created from nothing. This will be covered in Chapter 5.

Proof of Work: an Energy Intensive Assymetric Puzzle

The solution to these three problems is called Proof of Work. It was actually invented way before Bitcoin, in 1993.

We need to make it expensive to "buy tickets" to the lottery, otherwise people could generate an unlimited number of tickets. What's something that is provably expensive, but doesn't have to come from any central authority?

I did mention physics in the beginning of the book, and this is where physics plays into Bitcoin: the first law of thermodynamics says that energy can neither be created nor destroyed. In other words, there's no such thing as a free lunch when it comes to energy. Electricity is always expensive because it is a scarce resource which costs real money. You have to purchase it from the power producers, or run your own power plant. In either case, you can't get something from nothing.

The concept behind Proof of Work is that you participate in a random process, similar to rolling a die. But instead of a six sided die, this one has about as many sides as there are atoms in the universe. In order to roll the die and generate lottery numbers, your computer must perform many operations, which cost you in terms of electricity.

To win the lottery, you must produce a special number which is mathematically derived from the transactions you want to write to the ledger plus a random number (we'll explain the details of how this works in the next chapter).

In order to find this winning number, you may have to roll this die billions, trillions, or quadrillions of times, burning hundreds or thousands of dollars worth of energy. Since the process is based on randomness, it is possible for everyone to generate their own lottery tickets without a central authority using basically just a random number generating piece of software or hardware and a list of transactions they want to write to the ledger.

Now even though it may have taken you thousands of dollars to burn enough energy to find the correct random number, in order for everyone else on the network to validate that you're a winner, they need to perform two basic checks:

1. Is the number you provided less than or greater than the magic threshold everyone agreed upon?
2. Is the number indeed mathematically derived from a valid set of transactions that you want to write to the ledger?

This makes the Proof of Work system *asymmetric*: it is very hard to generate, but very easy to validate.

Because you've burned a considerable amount of money playing this lottery, you want everyone to accept your winning lottery ticket. Thus, you are incentivized to behave well by writing only valid transactions into the ledger.

If you, for example, try to spend money that's already been spent, then your "winning" lottery ticket will be rejected by everyone else, and you'll lose all the money you spent buying the energy to burn for the ticket. On the other hand, if you write valid transactions into the ledger, we'll reward you in bitcoin so you can pay your energy bills and keep some profit.

The Proof of Work system has an important property of being *real world expensive*. Thus, if you wanted to attack the network by coercing some of its participants, you'd have to not only come to their house and take over their computer, but you'd also have to pay their electrical bills.

Today, the Bitcoin network as a whole is estimated to expend more energy on this lottery than some medium sized countries. That's how much energy you would have to harness to cheat the network.

How do participants prove they've burned this energy? We'll discuss how Proof of Work is validated from a mathematical standpoint next.

4

BITCOIN MATH

Before we can discuss how Proof of Work is validated, we'll need a quick Computer Science primer on two concepts: bits and hashing.

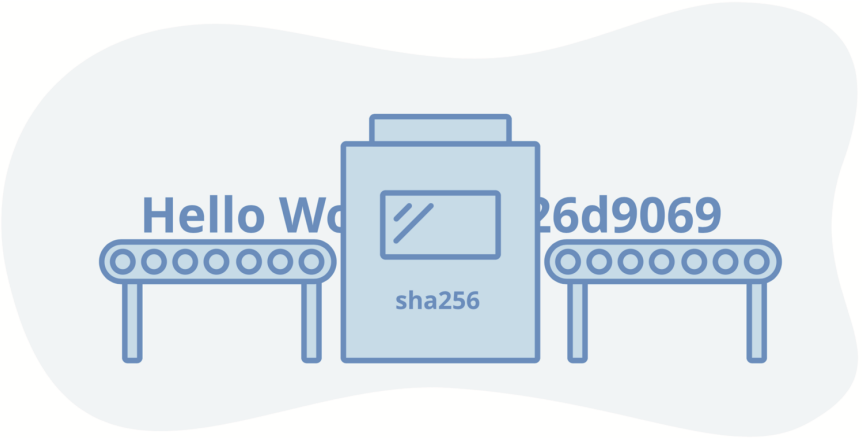
Hashing

Bitcoin's asymmetric Proof of Work puzzle involves using a *hash function*. From basic algebra, we know that a function is a box where you put in an *input* value x and you get an *output* value $f(x)$. For example, the function $f(x)=2x$ takes a value and multiplies it by two. So the input $x=2$ gives us the output $f(x)=4$.

A hash function is a special function, where you put in any string of letters, numbers, or other data, like "Hello world", and you get out a giant random looking number:

```
111811713258219242661329357757490458455
489044664361600112658434663354150209 5
```

The particular hash function I used to hash "Hello world" is called sha256 and happens to be the one Bitcoin uses.



The sha256 hash function has the following properties that are useful for us:

1. The output is deterministic: you always get the same output for the same input.
2. The output is unpredictable: changing just one letter or adding a space to the input string will drastically change the output, so much so that you cannot find any correlation to the original input.
3. It is quick to compute the hash for any size of input data.
4. It is basically impossible to find two strings that hash to the same output.
5. Given the output hash of sha256, it is impossible to arrive back at the input string. We call this a one-way function.
6. The output is always a specific size (256 bits for sha256).

A Quick Primer on Bits

The number system you know and love, comprised of the numbers 0 through 9 is called *decimal* because it has ten digits. Computers, on the other hand, prefer a different number system made of ones and zeros, indicating the presence or absence of an electrical signal. This number

system is called *binary*.

In the decimal system, you use only the digits 0 through 9. If you use only one digit you can represent ten different numbers, 0 through 9. If you use two digits, you can represent 10 x 10 = 100 different numbers: 00, 01, ... through 99. For three digits, you can have 10 x 10 x 10 = 1000 numbers: 000, 001, ... through 999.

Hopefully you're starting to see a pattern. To figure out how big a number we can represent with N digits, we multiply ten by itself N times, in other words 10^N, or 10 to the power of N.

Binary works the same way. The only thing that changes is the number of digits that are available to us. While we're used to decimal with ten digits, a *binary digit* or *bit* can only have two values: zero and one.

If 1 bit can represent two values, then two bits can represent 4 values: 00, 01, 10, 11. You can calculate this by multiplying 2 x 2 since each digit can have two values.

Three bits can represent 2 x 2 x 2 = 2^3 = 8 values, which are 000, 001, 010, 011, 100, 101, 110, 111.

So a *binary* number that is N *bits* long can represent 2^N different values.

Therefore, the number of unique values you can represent with 256 bits, the size of the sha256 hashing function, is 2^{256}. That's a giant, almost inconceivable number. Represented in decimal, this number is 78 digits long. To put this in perspective, it's in the same ballpark as the estimated number of atoms in the known universe.

2^{256} = 115,792,089,237,316,195,423,570,985,008,687,907,
853,269,984,665,640,564,039,457,584,007,913,129,639,936

This is the number of possible outputs when you hash any string with sha256 hash function. Thus, it is effectively impossible to predict what the number produced by this function will look like. It would be like

predicting 256 coin tosses in a row, or guessing the location of a specific atom that I've picked somewhere in the universe.

This number is too long to keep writing out, so we'll just say 2^{256} from now on, but I hope that this triggers a mental image of a universe of possibilities for you.

Let's Hash Some Strings

Here are some example strings and their sha256 hashes. I've shown their output as decimal numbers, though inside a computer these would appear as a binary string of ones and zeros.

The point here is to demonstrate how drastically the number changes based on a small change to the input string. You can't predict the output produced by the hash function based on what you put into it:

```
"Hello world!"
52740724284578854442640185928423074974
81806529570658746454048816174655413720

"Hello world!!"
95863319874939535731602344194643497258 3
74513872780665335270495834770720452323
```

There's no way for anyone, not even a computer, to look at the resulting random looking number and figure out the string that created it. If you want to play with sha256, there are places online where you can try it out.

Hashing to Win the Proof of Work Lottery

Alright, now we're ready to talk about the key bit of magic. We said there are 2^{256} total possible sha256 output values. To make it easier to

understand, let's pretend that there are only a total of 1000 possible hash outputs.

The lottery system works like this:

1. Alice announces she wants to send $2 to Bob.
2. Everyone playing the lottery takes this transaction "Alice Gives $2 to Bob", adding a random number called a *nonce* (number used only once) at the end. This is to make sure that the string they're hashing is different from anyone else, helping them to find a winning lottery number.
3. If that number is smaller than the *Target Number* (we'll get to this in a second), they win the lottery.
4. If the number they get is bigger than the Target Number, then they hash the same thing again, adding random nonces: "Alice Gives $2 to Bob nonce=12345", then "Alice Gives $2 to Bob nonce=92435", then "Alice Gives $2 to Bob nonce=13284901234809213 4", and so on, until the resulting hash number is smaller than the Target Number.

It may take many, many tries to find a hash that is less than the Target Number. So the idea here is this: if there are 1000 possible hashes, and we set the Target Number to 100, then what percentage of hashes are under the Target?

This is basic math: out of 1000 possible numbers, zero through 999, there are 100 numbers that are less than 100, and 900 numbers that are greater. Therefore 100/1000 or 10% of hashes are less than the Target. So if you hash any string and your hash function produces 1000 different outputs, then you're expecting to get a hash that's under the Target of 100 about 10% of the time.

This is how the lottery works: we set a Target, and we all agree on the Target (we'll talk about how that works in a bit). Then we all take the transactions that people have been telling us about, and hash them,

adding a random nonce at the end. Once someone finds a hash that's under the Target, we announce it to everyone on the network:

Hey everyone:

- I took the transactions: "Alice Sends $2 to Bob, Charlotte Sends $5 to Alice".
- I added the nonce "32895".
- It came out to an output hash of 42 which is less than the Target of 100.
- Here's my Proof of Work: the transaction data, the nonce I used, and the hash that was produced based on those inputs.

Now it might have taken me billions of tries of hashing to get there, burning thousands of dollars of energy, but everyone can immediately validate that I did it right because they can do the hash in one try since I gave them both the input and the expected output. Remember, hashes are impossible to reverse, but easy to compute!

How does this tie into energy burning? Well, we already said the set of all possible hashes is actually a giant number that's about as big as the number of atoms in the universe. Now we can set the Target to be low so that only a tiny fraction of hashes are valid. This means that anyone who wants to find a valid hash will have to spend a huge amount of computation time, and therefore electricity, to find a hash number smaller than our Target.

The smaller the Target, the more tries it will take to find a number that works. The bigger the Target, the faster we can find a winning hash.

5

MINING

Now we're ready to get into how the Proof of Work Lottery in Bitcoin actually works:

1. Anyone in the world who wants to participate, joins the Bitcoin network by connecting their computer and listening for transactions.
2. Alice announces her intent to send some coins to Bob. The computers on the network gossip with each other to spread this transaction to everyone on the network.
3. All the computers who want to participate in the lottery start hashing the transactions they heard about by appending random nonces to the transaction and running sha256 hash functions.
4. The first computer to find a hash number less than the current Target Number wins the lottery.
5. This computer announces the winning number they found, as well as the input (transactions and nonce) that they used to produce it. It might have taken them hours to get there, or a few minutes. This information taken together (transactions, nonce, and the Proof of Work hash) is called a *block*.

6. Everyone else validates the block by checking that the transactions in the block together with the nonce do indeed hash to what was claimed, that the hash is indeed lower than the Target Number, and that the block does not contain any invalid transactions, and that the history within it does not conflict with prior blocks.
7. Everyone writes the block into their copy of the ledger, appending it into the existing chain of blocks, producing a *blockchain*.

That's it. We've produced our first block and our first entry into our ledgers. The process of playing the Proof of Work lottery to win access to write to the Bitcoin ledger is known popularly as *mining.*

How are New Bitcoins Minted?

So far we discussed how Alice can send $2 to Bob. We're going to stop talking about dollars now, because Bitcoin doesn't know anything about dollars. What we do have are bitcoins themselves - digital units that represent value on the Bitcoin network.

To revisit our example, what's really happening is that Alice is sending 2 bitcoin to Bob by announcing that she's moving bitcoin that is registered under her "account" to Bob's. Someone then wins the Proof of Work lottery, and gets to write her transaction into the ledger.

But where did Alice get those 2 bitcoin to begin with? How did Bitcoin start, and how did anyone acquire Bitcoin before there were places to buy it for traditional fiat currency like the US dollar?

It turns out the process of mining bitcoin, which is the process of playing the Proof of Work lottery and getting access rights to the ledger, is the very thing that produces Bitcoin. When you find a valid Block (by burning a large amount of energy and finding a number that wins the

lottery), you get to write any transactions you've heard about into that Block and therefore into the ledger.

But you *also* get to write a very special additional transaction, called a *coinbase transaction* into the ledger. This transaction basically says: "12.5 Bitcoin were minted and given to Mary the Miner to compensate her for expending all that energy to mine this block."

The Block Reward

So the person who wins the lottery gets to give themselves some newly minted Bitcoin. Why is it 12.5 Bitcoin and not 1000? Why can't she cheat the system and give herself any amount? Here's the key part: Bitcoin is a system of distributed consensus. That means everyone has to agree on what is valid.

If Mary mines a block and decides to give herself a little something extra, everyone else's computer will *reject* this block as invalid, because inside of the Bitcoin client software that everyone is running, there's a piece of code that says "the current Block Reward is exactly 12.5 bitcoin. If you see a block that grants someone more than that, do not accept it."

If Mary tries to cheat and produce an *invalid* block, the block won't get written to anyone's ledger, and instead she will just have wasted thousands of dollars of electricity producing something no one wants - a forgery.

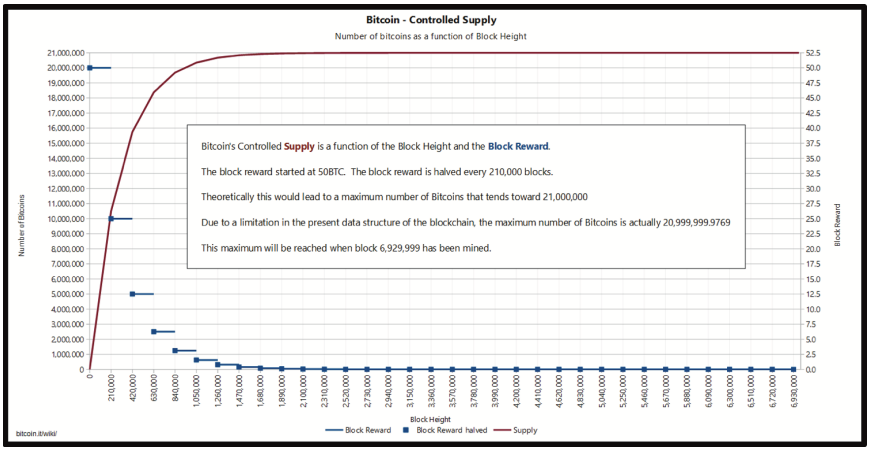
The very first block ever mined was mined by Satoshi. The code is open source - that means anyone could take a look at how it works and validate that nothing fishy is going on under the hood. Even Satoshi had to run computations and play the Proof of Work lottery to mine the first block.

In the beginning, the Block Reward was 50 bitcoin, so that's what Satoshi was rewarded for mining the first block, as did the other people who joined the network in the early days and mined the first blocks.

The Bitcoin code enforces a Block Reward Halving roughly every four years. This is based on the number of blocks mined, rather than the passage of time, but they are almost the same due to blocks being produced roughly every ten minutes.

The Block Reward in 2008 was 50, in 2012 was 25, in 2016 was 12.5. As of today, Jan 15, 2019 - there have been 558,688 blocks mined since the beginning of Bitcoin's history, and the reward is 12.5 bitcoin per block.

71,312 blocks from now, or approximately in late May, 2020 the reward will be lowered to 6.25 Bitcoin per block, leading to an annual supply increase rate of approximately 1.8%. A decade later, following two more reward halvings, more than 99% of all Bitcoin will have been mined and less than 1 bitcoin will be produced per block.



https://en.bitcoin.it/w/images/en/4/42/Controlled_supply-supply_over_block_height.png

Eventually, around the year 2140, the Block Reward will go away entirely, and miners will be incentivized only by fees paid by those performing the transactions.

These issuance and Block Reward numbers are enforced in the Bitcoin

code—which, to reiterate, is completely open source and can be validated by anyone—so depending on how far along we are in Bitcoin's history, producing a block that doesn't follow these rules will get you rejected by everyone else who is checking the same rules written into their code.

Controlling the Issuance and Mining Interval

Mining requires computing hardware and electricity, so the more hardware and electricity you control, the more likely you are to find the winning number faster than other people. For example, if there are 100 equally powered computers on the network, and you control 10 of them, then you will find the winning block *approximately* 10% of the time. However, mining is a process based on chance and randomness, so it is possible that hours or even days can go by without you ever finding a block.

We know from the prior section that miners can't just grant themselves arbitrary block rewards, or they would get rejected by the other nodes. But what if they burn a whole bunch of energy to speed up mining blocks and get their hands on a whole lot of bitcoins, violating the design constraint that the issuance schedule should be known in advance?

Let's again go to the example of there being only 1000 possible hashes and our Target Number being 100. That means 10% of the time we will roll a number that's less than 100 and find a block.

Let's say it takes us 1 second to compute each hash. If each second we "roll our die" by hashing the current transactions and our random nonce, and 10% of the time we'll hit a number less than the Target, then we expect it will take us about 10 seconds on average to find a valid hash.

What happens if two computers are playing the lottery? They're hashing twice as fast, so we'll expect a valid hash to be found within 5

seconds. What if 10 computers are playing? One of them will find a valid hash approximately every second.

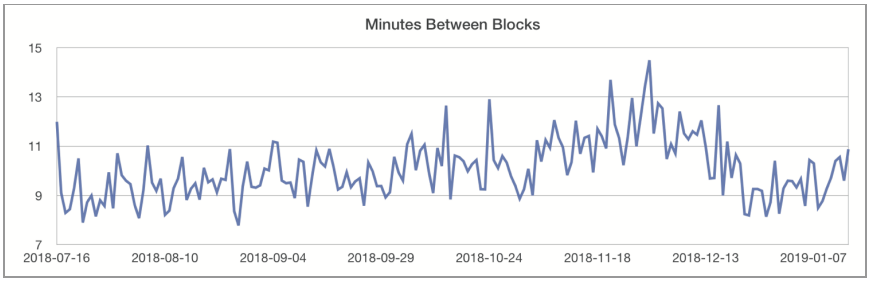
This creates a problem: if more people are mining, then blocks will be produced too quickly. This has two outcomes that we do not want:

1. It interferes with the idea of having a predetermined issuance schedule. We want a relatively consistent number of bitcoins per hour to be issued in order to make sure we issue all of them by the year 2140, and not any time ahead of that.
2. It creates networking problems: if blocks are mined so quickly that they don't have time to reach the entire network before the next one is mined, then we cannot come to consensus on a linear history of transactions, since multiple miners may include the same transaction in their blocks, leading to blocks being invalid because they contain transactions that were already spent in other blocks.

And if fewer people are mining, we create the opposite problem:

1. Bitcoins are being emitted too slowly, again interfering with issuance.
2. The blockchain may become unusable as people wait hours, days, etc, to get a transaction written to the ledger.

The total number of hashes per second performed by all the miners of the Bitcoin network is referred to as the *hash rate*.



Difficulty Adjustments: Agreeing on the Target

How can we make it harder to find valid hashes if more players join the lottery and easier if players leave the lottery, in order to keep the issuance and block times steady?

Bitcoin solves this problem with a *mining difficulty adjustment*. Because everyone is running the same code, which enforces the same rules, and everyone has a copy of the entire history of blocks to this point, everyone can independently calculate how fast blocks are being produced.

Every time we produce 2016 blocks, which is roughly equivalent to two weeks of time, we look back and figure out how long it took us to produce those blocks, and then adjust our Target Number to speed up or slow down block production.

Everyone takes the last 2016 blocks and divides them by the time they took to produce to create an average. Did it come to more than ten minutes? We're going too slow. Did it come to less than ten minutes? We're going too fast.

Now we can make an adjustment to the Target Number so that it is raised or lowered proportionally to how much faster or slower we want to go based on the 10 minute interval which is written into the open source code.

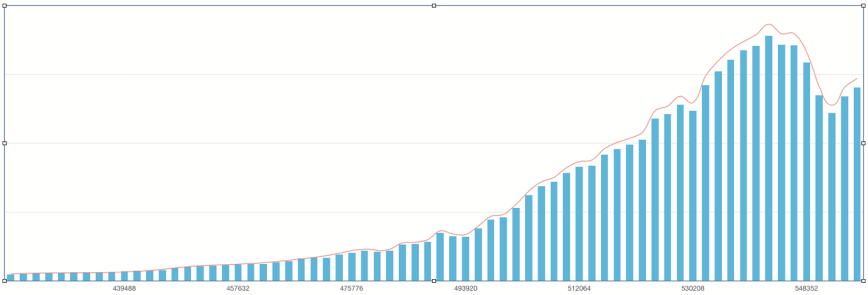
We can raise the Target Number to a higher number, creating a larger

space of valid hashes, giving miners a higher chance to find a winning hash, thus expending less energy. This is called *lowering the difficulty*. Alternatively we can lower the Target Number so fewer hashes are valid and miners have to spend more energy finding a valid hash. This is called *raising the difficulty*.

This also means that for any block, based on how many blocks have come before it (its *block height*), we know exactly what the Target Number is. That lets us know the magic threshold under which the Proof of Work hash number has to fall for a winning lottery ticket for that specific block.

This is brilliant - we no longer need a central party to tell us anything. All we need to do is verify for ourselves what the Target should be and whether the Proof of Work claimed by a lottery ticket is a winning number falling under the Target.

The chart below shows the hash rate as a line, and the difficulty as bars over time. The difficulty looks like a staircase because it is adjusted in 2016 block increments. You can see that every time the hash rate rises above the difficulty, the difficulty steps up to catch up to the hash rate. When the hash rate falls, as it did between Oct-Dec of 2018, the difficulty steps down. The difficulty adjustment always lags behind whatever the hash rate does.



Hash Rate vs. Difficulty

Because there is a 2016 block lag for difficulty adjustment, it is possible

for massive spikes up or down in hash rate to over or under produce Bitcoin during that 2016 block period, and slightly violate the issuance schedule. In fact, we are running a bit fast right now compared with the original target of finishing issuance in 2140. Since adding hash rate typically means producing a large quantity of new hardware, this is relatively unusual and doesn't impact things too greatly, and we expect it will likely even out in the future.

We've almost completed inventing all of Bitcoin:

1. Replaced a central bank with a distributed ledger.
2. Instituted a lottery system to select who writes to the ledger.
3. Forced lottery entrants to burn energy to buy tickets by hashing, and made it easy for everyone to verify winning tickets by checking the hash numbers produced by players.
4. Told the lottery players that if they didn't play by the rules, we'd reject their blocks including their *coinbase transactions* so they wouldn't get paid when they won, thus creating an economic disincentive for cheating, and an economic incentive for playing by the rules.
5. Controlled the timing and Target selection for the lottery by letting everyone calculate for themselves what the Target should be based on hardcoded rules and the history of the past 2016 blocks
6. Enforced the issuance schedule using difficulty adjustments that adjust to increasing or decreasing hash rate.
7. Used open source code to ensure that everyone could verify for themselves that they were enforcing the same rules regarding transaction validity, block reward, and difficulty calculation.

No more central party. We have a fully distributed and decentralized system. Anyone can join. Anyone can play the lottery and mine bitcoins for themselves. Anyone can spend. Honest production of blocks is validated by the entire network, and rewarded with a coinbase transaction

that pays the miner, or punished by lack of reward and the miner having expended energy to mine.

We've almost got the entire picture. There remains one problem. When someone joins the network and asks for copies of the ledger, they may get different ledger histories from different nodes. How do we enforce a single, linear history, and how can we prevent miners from rewriting the past?

6

SECURING OUR COINS

So far we talked about how we manage keeping copies and writing to a distributed ledger without allowing for coercion or corruption. But what happens when a lottery winner wants to get malicious? Does winning the right to write to our ledger mean they can change historical entries in the ledger? Can Eve, Dave, and Farrah collude in order to rewrite history or change account balances and give themselves extra coins?

Enter the blockchain. A marketing term that has permeated much of the tech sector, the blockchain is nothing more than the idea that Bitcoin blocks are chained together to provide links from one set of transactions to the next.

We lied a little in the prior chapter to keep things simple. When you mine by playing the Proof of Work lottery, you're not only hashing the transactions that want to go into the next block along with nonce. You're actually also providing the hash of the block that came before yours as input into your hash function, thereby linking your block to the prior block.

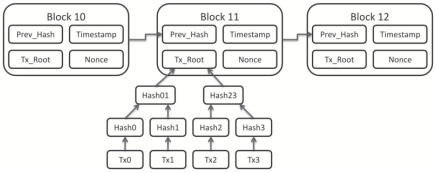
This enables us to build a historical record of every block back to the

very first Genesis Block mined by Satoshi. When we write a new block into the chain, we have to validate that this block does not contain any transactions that spend bitcoins that were already spent in prior blocks.

Recall that the output of a hash function is random and dependent on all the data input into it. We've now modified our block hashes to include three different inputs:

1. The transactions we want to commit to the ledger
2. A random nonce
3. The hash of the prior block that we're using as the basis of our ledger's history

If any of those three things change, the output hash changes in an unpredictable and drastic way. This behavior creates an interesting property: if you tamper with data in any old block, you will change its hash. If you change the hash of any old block, you will change the hash of every single block that comes later.



https://upload.wikimedia.org/wikipedia/commons/7/7a/Bitcoin_Block_Data.png

This makes the chain *tamper evident*. If anyone were to try to change the an older block in the chain, they would have to recompute the Proof of Work hash of the block they're tampering with, and every single block that comes afterwards.

Effectively, every new block that is mined in Bitcoin adds to the security of the blocks that came before it. A transaction in a block that is buried

under 6 blocks is considered as good as set in stone. It would take a tremendous amount of energy to recompute the last six blocks at today's total hash rate. One that's 100 blocks deep? Forget about it.

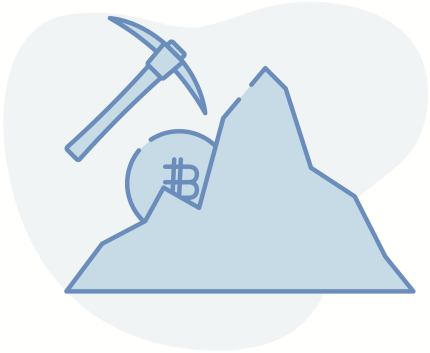
It's important to understand that there is no actual transaction finality in Bitcoin. Every merchant or payment processor decides for themselves what they consider final. Today, most people accept 6 confirmations—six blocks mined on top of the one which contains a transaction—as final, but merchants can set this as they wish.

If you're selling a digital book that has marginal cost to you as a merchant, you might only want 1 confirmation, or even zero confirmations, delivering the digital good as soon as you see the transaction broadcast on the network. If you're selling a house, maybe you want to wait for twelve confirmations, or roughly two hours of mining. The longer you wait, the more Proof of Work is piled on top of the block that contains your transactions, and the more real world expensive it becomes to reverse the transaction.

If the hash rate of Bitcoin were to fall significantly, meaning that less energy was securing each block, one could always increase the number of confirmations they would require for final settlement. Although this may seem disconcerting at first, it's important to keep in mind that credit card transactions can typically be reversed 120 days after they are made. On the other hand Bitcoin is a final settlement money that cannot be taken from you, like cash or gold. From this standpoint, the reversibility and finality of Bitcoin transactions is actually a vast improvement to that of most traditional payment networks.

Let's go back to our example from Chapter 3 where Henry joins a network and gets different copies of the ledger. The ledger he gets from Charlotte is honest, but the ledgers from Eve, Dave, and Farrah are malicious, where they've deleted an old block that contained Alice's original spends so that they could fool Henry into thinking she still had her coins. Prior to having the blocks linked by Proof of Work to each other, Henry couldn't know that an old block was deleted.

Because each block contains Proof of Work, he knows roughly how much energy was burned to produce it based on the Target Number. Because each block points to a prior block, he knows that to change the history of an old block would require recomputing the Proof of Work not just of the tampered block, but every block that came after it. Because he can see all the transactions in each block, he can ensure no double spends were made.



Unlike mining gold, which also burns energy, the process of Bitcoin mining actually secures the network to make the ledger tamper proof.

What if two people find a block at the same time?

There is one missing piece of the consensus system. Imagine we are now running a worldwide network. People across the world, from the U.S. to China are connected to this global network and they're all playing the Proof of Work mining lottery.

Someone in Chicago finds a valid block. They announce it to the network and all the computers across America start picking it up. Meanwhile, someone in Shanghai also finds a block within a few seconds of the Chicago block. Their neighbors still haven't heard about the American block, so they hear about the Chinese block first.

As the two blocks propagate through neighboring nodes on the network, we now have two competing versions of the blockchain. The

Americans have one that has the American block at the end, and the Chinese have their own block. The network is split on which blockchain is the correct copy of the ledger since both contain equivalent amounts of Proof of Work and both contain valid transactions.

You can't rely on any central party to tell you which one wins. What do you do? Bitcoin provides for a simple solution here: let's just wait and see. There are now two competing versions of the blockchain. In the next roughly ten minute period, another block will be mined. The Americans will be mining on top of the block they first heard about, and the Chinese will be mining on top of their block.

Whichever side mines the next block will win. How? Because in the Bitcoin code there is a rule that says that the Longest Cumulative Proof of Work chain resolves any chain splits. Whoever expends the most energy wins. The rule of resolving inconsistencies between chains based on their total cumulative Proof of Work is now called Nakamoto Consensus, in honor of Satoshi Nakamoto.

Let's say the Chinese mine the next block. Their chain is now one block ahead of the American one. When they broadcast this finding, the American bitcoin nodes will recognize that the Chinese nodes have produced a longer cumulative Proof of Work chain, and reorganize (or "reorg"). That means they will throw away the block they mined in favor of the two Chinese ones. The American block is now called an *orphan*. Since it was rejected, it means the miner who mined it didn't get rewarded.

You may notice that even though I've referred to nodes as American and Chinese, in reality the nodes don't know anything about each other's identity, geographic location, and so on. The only proof of validity they need is that someone has the Longest Cumulative Proof of Work chain, and that the transactions in the chain are themselves all valid (not double-spends, etc).

The probability of this chain split is very low — it used to happen once

a month or less, but recently hasn't happened much at all due to improvements in block propagation technology and networking connectivity between miners.

Part of the reason Bitcoin produces relatively small blocks roughly every ten minutes is to ensure that orphans are extremely rare. The other reason is to keep the hardware requirements for running a node relatively low to encourage more nodes in the system.

If we produced blocks every second or had very large blocks, we would have a very high probability that American and Chinese blocks would conflict because they're far apart geographically and take longer to reach each other. If orphans happen too often, then the blockchain would unravel because there would be orphans upon orphans and nodes would not be able to agree on a linear history of transactions.

A Bitcoin node needs only needs to connect to one other honest node that has the entire latest blockchain on the network in order to avoid being fooled by attackers who might feed it false information. Nodes constantly gossip with each other to make sure they have the latest blocks. If your node wants to know which copy of the blockchain is true, it only needs to look for the chain with the most Cumulative Proof of Work. Since everyone else is also following this rule, hardcoded in the software, it ensures that there is consensus on what the true state of the ledger is.

It is extremely difficult, therefore, for malicious hackers to give a node a false copy of the blockchain, as it would require severing that node's connection to any other honest nodes, and connecting it only to evil nodes.

Although forking typically happens due to chance and block propagation delays rather than malice, it is also possible that a malicious entity that wants to control what goes into the next block can take advantage of Nakamoto Consensus by controlling more than 50% of the total hash rate of the network and producing the longest Cumulative Proof of

Work chain. We'll discuss the details of this so-called "51% attack" in Chapter 9.

Security and the Dollar Value of Bitcoin

We determined that Bitcoin automatically recomputes difficulty based on the number of lottery players, that is miners expending energy through hashing. Here's where the real world starts to touch our digital world. The price of Bitcoin, the price of hardware and energy, and the difficulty value create complex feedback loops:

1. Miners produce bitcoin by spending money on energy because they think it will have some value.
2. Speculators buy bitcoin because they think it's going up, driving up the price to $X.
3. Miners spend up to $X of energy and hardware to try to mine a bitcoin.
4. A high demand from buyers and a rise in price drives more miners to mine bitcoin.
5. More miners means more energy spent on bitcoin and the network gets even more secure, reassuring buyers of Bitcoin's security, sometimes leading to a feedback loop to drive price even higher.
6. After 2016 blocks pass, the presence of more miners and thus more hash rate causes an upwards difficulty adjustment.
7. A larger difficulty means a lower Target Number—miners are finding blocks less often—causing at least some of them to spend more than $X in operating costs to mine a coin.
8. Some miners become unprofitable, spending more energy to mine than they can sell the bitcoin for. They turn off their miners.
9. Another 2016 blocks pass. The difficulty is recomputed to become easier, since some miners went offline.
10. A lower difficulty means that miners that were previously

unprofitable can come back online and mine, or new miners can join the game.

II. Go to I.

In a downward market, the cycle can go in the other direction, with users dumping coins, causing the price to go down, and miners to become unprofitable. Yet, contrary to what you might read in the media about a "death spiral," the difficulty adjustment algorithm ensures that there will always be some type of equilibrium between the price and the number of miners on the network. It also pushes out inefficient miners in favor of ones operating on the cheapest possible energy.

In practice over the last few years, the price has climbed very quickly, as has the total hash rate. The higher the hash rate, the harder it is to attack the network because in order to control what gets written to even just the next block, you'd need to have as much energy and hardware under your control as more than half of the entire network. Today, the energy expended by the network of Bitcoin miners is estimated as equivalent to that of a medium sized country.

7

ACCOUNTS WITHOUT IDENTITY

So far we have built a distributed ledger with no central authority, a lottery system for selecting who writes to it, a system for rewarding good miners and punishing misbehaving ones, a way to adjust mining difficulty to ensure a consistent issuance schedule and reduce conflicts, and a system for checking the validity of the chain by looking at the cumulative proof of work and transaction history.

Now let's deal with identity. In a traditional banking system, you send money by identifying yourself to the bank, whether through presenting an ID and pin code in person, or by presenting a username/password in an app. The bank ensures that no two entities share an identity.

Since we now have no central party to keep track of identities, how can we open accounts in our new Bitcoin based financial system, and how can we ensure that when Alice announces she wants to pay Bob, that it's really her and that she has authority to move those funds?

Generating a "Bitcoin Account"

Since we can't rely on a central middleman like a bank to keep a record

of all accounts, and because people can come and go as they please without permission, how can we manage accounts?

What if we let everyone register their own username/password? A bank would normally check that a username isn't already in use, but that's not possible here, since we have no central actor handing out identities. So we need something bigger and stronger, and more unique, than a username and password. This technique should be familiar from prior chapters. We once again need a giant random number.

Just like we made it possible for everyone to buy lottery tickets by generating large random numbers, we can use the same trick for generating accounts. To create a "Bitcoin Account," also known as an address, we will first generate a pair of 256 bit numbers that are mathematically linked, known as a *public/private key pair*. Again, 2^{256} is as large as the number of atoms in the universe, so two people accidentally generating the same key pair is next to impossible.

The key pair has some interesting properties. You can use either key to encrypt a message, and the other to decrypt it. Additionally, you are welcome to share your public key with the entire world. Knowing that key does not allow them to have access to your private key.

Let's take a look at how Alice sends coins to Bob. To receive a transaction, Bob generates a private/public key pair, and keeps his private key secret. He produces an *address*, a large number based on his public key. Bob then shares this address number with Alice so she can send him coins.

Alice now has to tell the network that she is sending coins to Bob's public address from her own public address. How does she prove that she is authorized to spend from that public address? She does this by providing proof that she has the private key to that address, but without actually revealing her private key.

This proof is called a *digital signature*. Alice constructs a transaction, which essentially just a piece of data that looks something like "address

12345 is sending 2 bitcoins to address 56789," except the address numbers are actually giant numbers. She then hashes her transaction and encrypts the hash with her private key, creating a digital signature.

When she publishes her transaction to the network, she reveals her public key (where she's sending from). Since everyone has the public key they can easily decrypt the digital signature. But they would only be able to decrypt it if it was in fact encrypted with the corresponding private key that only Alice knows.

Therefore, just the virtue of being able to correctly decrypt the signature allows everyone to know that Alice has the private key to that address, without actually revealing her private key.

When you move money in a bank, you give them your username and password. When you write checks, you sign your name to authenticate that it's you writing the check. When you move bitcoin, you provide proof that you own the key to the address that holds the bitcoin.

Unlike a signature on a check or your bank password, however, your digital signature is specific to the unique transaction data you are signing. Thus it can't be stolen and reused on a different transaction. Every transaction gets a different signature, even if it's based on the same private key.

Can You Guess a Private Key?

Let's figure out the odds of guessing a private key, which would give you the ability to move the coins at the corresponding public address. Remember, a key is made up of 256 bits. Each bit has only two values (one or zero). That means you can visualize each bit like a coin toss.

If we had a 1-bit private key, it's like tossing a coin. Heads or tails, one or zero? You have a one in two chance of guessing right.

Quick basic probability review: the probability of multiple events occurring is calculated by multiplying together each event's individual

probability. If a coin toss has a 1/2 chance of landing heads, then the chance of two coin tosses in a row landing heads is 1/2 x 1/2 = 1/4 or 1 in 4.

If we had 2 bits, that's two coin tosses in a row. $2^2 = 4$, so you have a one in 4 chance.

If you were to guess the outcome of 8 coin tosses in a row that would be 2^8, or a one in 256 chance.

A license plate has 6 letters/numbers. There are 26 letters and 10 numbers, so a total of 36 characters. Since there are six of them, the number of possible license plates = 36^6, so your odds of guessing mine are one in 2,176,782,336 (one in two billion)[1].

A credit card is sixteen digits. Each digit can have 10 values, and there are 16 of them so your odds of guessing my credit card are one in 10^{16}, which is one in 10,000,000,000,000,000 or roughly one in ten quintillion.

There are about 10^{50} atoms on earth. If I'm thinking of one at random, your chances of guessing it are about

> One in 1,000,000,000,000,000,000,000,000,000,000,
> 000,000,000,000,000,000,000,000,000.

A private key has 256 bits, which is 2^{256} or about 10^{77}. It's actually closer in magnitude to guessing a specific atom from the entire universe, or winning the Powerball Lottery 9 times in a row:

> One chance in 115,792,089,237,316,195,423,570,985,
> 008,687,907,853,269,984,665,640,564,039,457,584,
> 007,913,129,639,936

But what if you had a super powerful computer to do the guessing? I can't do this subject justice more than this Reddit post[2], which I recommend reading in its entirety. While it gets technical, the final paragraph

gives you a good idea of what it would take to list all possible 256-bit keys:

> *"So, if you could use the entire planet as a hard drive, storing 1 byte per atom, using stars as fuel, and cycling through 1 trillion keys per second, you'd need 37 octillion Earths to store it, and 237 billion suns to power the device capable of doing it, all of which would take you 3.6717 octodecillion years."*
>
> — u/PSBlake on r/Bitcoin

Basically, it's impossible for you to guess someone's private key. Not only that, but the number of Bitcoin addresses is so large, that Bitcoin best practices actually call for generating a new address for every transaction you make. So instead of having one bank account, you might have thousands or even millions of Bitcoin accounts, one for every transaction you've ever received.

It may be disconcerting that your Bitcoin account is secured only by chance, but hopefully the illustration above gives you an idea that this is vastly more secure than the password to your bank account, stored on a centralized server, available to hackers.

Tracking Balances

It's time to correct one final white lie we've told about how Bitcoin works so far. There actually aren't any balances kept in the ledger. Instead, Bitcoin uses a model called UTXO: Unspent Transaction Outputs.

The idea of UTXO is that each transaction is a set of inputs that are consumed to produce new outputs. Think of it like sending a bunch of coins into a machine that melts the down and mints new coins of any denomination we want. A UTXO is simply the word for a transaction output—a coin produced by a prior transaction, including a *coinbase*

transaction from a block reward—that hasn't yet been spent to another address.

Let's say Alice has an address that contains 1 bitcoin. She wants to send 0.3 bitcoins to Bob. She generates a transaction that shows her address with a 1 bitcoin UTXO as an input and two outputs: a new bitcoin UTXO worth 0.3 as an output to Bob's address, and a new bitcoin UTXO worth 0.7 as an output back to her own address as change. The change can go to her original sending address, or for better privacy, she can send it to a new address that she generates on the fly.

Since there's no way on the chain to tell who controls which address (for that, you'd need to know the corresponding private keys and tie them to real world identities), the UTXO model enables a very nice privacy mechanism by enabling the creation of new addresses every time coins are moved.

Thus, to check the "balance" of a particular address, we actually have to add up all the UTXOs that have this address as an output. The total set of current UTXOs in Bitcoin grows when people send from one address to many, and shrinks when people perform "consolidation" transactions where coins from many addresses are spent to one address.

The UTXO model allows for easy and efficient validation of double spends, since any particular UTXO can only be spent once. We do not need to know the entire history of spends from a particular account.

We can also create and destroy any number of UTXOs at once, creating complex transactions that mix different inputs and outputs. This allows for the idea of "coin mixing" where multiple parties participate in a single Bitcoin transaction that mixes any number of inputs to produce any number of outputs, thus obscuring the history of the UTXOs. It also allows people to consolidate coins from many addresses to one, or spread them amongst multiple addresses to improve security and privacy.

Wallets

Since generating an account is nothing more than generating a random 256 bit number to be your private key, and we can create thousands or millions of accounts, we need a mechanism for tracking them. In Bitcoin, the word *wallet* is used to refer to any kind of device that tracks your keys. It could be as simple as a piece of paper or as complex as a piece of hardware.

The original Bitcoin software published by Satoshi came with a software wallet. This wallet would generate your private/public key pair (recall again that the public key is used to create your Bitcoin address, and your private key allows you to sign for transactions to spend coins from that address).

Unlike your bank's wallet, which is typically in the form of a single mobile or web application, Bitcoin is a completely open system. Thus there are hundreds of wallets, most of which are free, many of which are also open source, as well as a half dozen hardware wallet implementations with more coming. Anyone with knowledge of computer programming can build their own wallet or read the code of an open source wallet to ensure nothing fishy is going on. This is another place in Bitcoin where permissionless innovation is happening at a rapid clip, unlike your bank's mobile app.

Since your private key is the only thing you need to spend your coins, you must guard it very closely. If someone steals your credit card, you can call up the company and file a fraud complaint and try to get your money back. In Bitcoin, there is no intermediary. If someone has your private key, they control your coins, and there's no one you can call.

Private keys are also highly susceptible to loss. If you store your wallet on your computer and the computer is stolen or catches fire, you have a problem. If you follow Bitcoin best practices in generating a new address every time you receive payment, then securely storing and backing up these private keys becomes quickly burdensome.

Over time, the Bitcoin ecosystem has evolved a number of solutions to this problem. In 2012, BIP32 (Bitcoin Improvement Proposal, a mechanism for people to spread ideas on how to improve Bitcoin) was filed with the proposal to create Hierarchical Deterministic Wallets. The idea behind this is that using only a single random number (seed), we can generate an entire chain of private/public key pairs: Bitcoin addresses and signature keys for them.

Nowadays, if you use any of the commonly available software or hardware wallets, they will automatically generate new keys for you for every transaction, and allow you to back up only a single seed.

In 2013, BIP39 came along to make key backup even easier. Instead of using a completely random number, keys would be generated from a random set of human readable words instead. Here's an example seed:

```
witch collapse practice feed shame open
despair creek road again ice least
```

With this method, backing up keys became very easy: you could write the seed on a piece of paper and put it into a safety deposit box. You could even memorize the phrase and walk out of a failing economic regime like Venezuela with nothing on your person, no one being the wiser that you're carrying your wealth in your head.

Furthermore, a Bitcoin address can require more than one private key to access. Multisignature or *multisig* addresses may employ a large variety of security schemes. For example, two people can share an account using 1-of-2 multisig where either party can sign for transactions, or a 2-of-2 multisig that requires both parties to supply keys to spend.

You can make a simple escrow system using a 2-of-3 multisig. The buyer gets one key, the seller gets another key, and a third key is given to an arbitrator. If buyer and seller agree, they can unlock the funds together.

In the case of dispute, the arbitrator can act in concert with one of the parties to unlock the funds.

You can use a 3-of-5 multisig scheme to protect yourself from loss of keys by allowing yourself to lose up to 2 of the 5 keys and still being able to unlock the account. You can store two of the keys in different places, two with different trusted friends that don't know each other, and one with a specialized custodian service like BitGo which co-signs your transactions, making your Bitcoin very difficult to steal while protecting yourself from loss of keys.

You can go even further and make addresses that are unlocked by rather complex conditions, such as knowledge of secret numbers, or being locked for some specific period of time. You can, for example, make a bitcoin address from which you can't spend for 10 years, no matter how much anyone wants to force you to change it.

This is profound and world-changing. Never before has it been possible to carry your wealth in a way completely safe from seizure or theft.

---

1. The inspiration for this section came from an excellent Medium post which details the probabilities of a variety of events. I recommend reading the full post for context: https://medium.com/@kerbleski/a-dance-with-infinity-980bd8e9a781
2. The full Reddit post describing guessing a 256bit key is available here: https://bit.ly/2Dbw9Qd

8

THE BITCOIN CLIENT SOFTWARE

So now we've got a functional distributed system for keeping track of and transferring value. Let's review what we've created so far:

1. A distributed ledger, a copy of which is kept by every participant.
2. A lottery system based on Proof of Work and difficulty adjustments to keep the network secure and the issuance schedule consistent.
3. A consensus system that ensures that every participant can validate the entire history of the blockchain for themselves using an open source piece of software called the Bitcoin client.
4. An identity system using digital signatures that allows the arbitrary creation of account-like mailboxes that can receive bitcoins without a central authority.

Now it's time to tackle one of the most interesting and counterintuitive things about Bitcoin: where its rules come from and how they are enforced.

The Bitcoin Software

Throughout the prior chapters, we assumed that everyone on the network was validating the same rules: that is, they are rejecting double-spends, ensuring that every block contains the appropriate amount of Proof of Work, and that each block points to the prior block at the tip of the current blockchain, among a whole bunch of other things that people have agreed to over time.

We also said Bitcoin is an open source piece of software. Open source means anyone can read its code, and also that anyone can update their own copy with whatever code they want. How do changes make it into Bitcoin?

Bitcoin is a *protocol*. In computer software, this term refers to a set of rules that the software follows. However, as long as you follow the set of rules that everyone else is following, you are free to modify your software as you wish. When we say that people "run Bitcoin nodes," what we really mean is they run software that speaks the Bitcoin protocol. This software can communicate with other Bitcoin nodes, transmit transactions and blocks to them, discover other nodes to peer with, and so on.

The actual details of how the software is implemented is up to anyone running the software. In fact, there are many implementations of the Bitcoin protocol. The most popular of these is called Bitcoin Core, and is the extension of the work first released by Satoshi Nakamoto.

There are other clients as well, some even written in other computer languages and maintained by different people. Because consensus in Bitcoin is critical (meaning all nodes must agree on which blocks are or are not valid), the vast majority of nodes run the same software (Bitcoin Core) in order to avoid any incidental bugs that may cause some nodes to disagree on what is valid.

So who makes the rules?

The rules that make up Bitcoin are encoded into the Bitcoin Core client. But who decides these rules? Why do we say that Bitcoin is scarce if someone can come in and make a modification to the software that changes the 21 million bitcoin limit to 42 million?

Being a distributed system, all the nodes in the system must agree to the rules. If you're a miner and you decide to change your software to grant you twice as much Bitcoin as you're allowed by the current Block Reward setting, then when you mine your block, every other node in the network will reject your block. Making a change to the rules is extremely hard because there are thousands of nodes distributed across the world, each enforcing the rules of Bitcoin.

Bitcoin's governance model is counterintuitive, especially to those of us living in a western democracy. We are used to governance by voting - the majority of people can decide to do something, pass a law, and impose their will on the minority. But Bitcoin's system of governance is much closer to anarchy than democracy. Let's take a look at the checks and balances in this system:

Nodes: each participant in the Bitcoin network runs a node. They choose which software to run on this node. While most people run Bitcoin Core, if the software became malicious and tried to introduce something like inflation, then nobody would run it. Examples of nodes include those run by anyone who accepts Bitcoin - merchants, exchanges, wallet providers, and everyday people using Bitcoin for whatever purpose they want.

Miners: some nodes also mine. This means they expend electricity in order to win rights to write to the Bitcoin ledger. This provides the security of the network by making it very expensive for someone to tamper with the ledger. If the miners are the only ones who write to the ledger, it might be tempting to think of them as the rule makers, but

they are not. They are simply following the rules set by the nodes that accept bitcoins. If the miners start producing blocks that contain extra reward, they will be not be accepted by other nodes, thus leading those coins to be worthless. Thus, every user running a node is participating in anarchic governance - they are choosing which rules the coins that they consider Bitcoin should follow, and any violation of these rules is rejected outright.

Users/Investors: users are the people who buy and sell the bitcoin currency as well as run nodes. Many users today don't run their own nodes, but rely on a node hosted by their wallet provider, where the wallet provider acts as a sort of proxy for the wants and desires of the user. Users decide the value of the coin on the open market. Even if the miners and most of the economic nodes in the system were to collude and introduce some kind of radical change such as inflation, users would likely dump the currency, driving the price low and putting the offending companies out of business. An intolerant minority of users could always keep their own version of Bitcoin alive even if Bitcoin morphed into something they didn't like.

Developers: the Bitcoin Core software is the largest Bitcoin client project out there. It has attracted a rich ecosystem of hundreds of the best crypto developers and companies. The core project is very conservative as the software powers a network that now secures hundreds of billions of dollars. Each change goes through a proposal process and is carefully peer-reviewed. The process for proposals and code review is done completely in the open, and anyone can join, comment, or submit code. If the developers become malicious and introduce something that nobody wants to run, then users would simply run different software (perhaps older versions, or start developing something new). Because of this, the core developers must develop changes that users would generally want, or risk losing their status as the reference implementation if no one wants to run it.

The Bitcoin ecosystem is a delicate dance between hundreds and thousands of participants, all of which are acting selfishly, and often with competing needs, but producing a highly resilient system for the greater good as a result. It's a truly free-market anarchist system with no one in particular in charge.

9

PAST, PRESENT, AND FUTURE

Now armed with an understanding of the Bitcoin network as a whole, we can examine a few interesting behaviors that have emerged over the last ten years of the system.

ASICs and Mining Pools

In the beginning, Satoshi mined the first bitcoins using his computer's central processing unit (CPU). Since the initial mining difficulty in the system was set low, it was relatively inexpensive for his computer to generate these coins.

Over time, people started tweaking the mining software to make it more and more efficient. Eventually, they wrote software that started taking advantage of specialized processors called graphics processing units (GPUs) which exist on graphics cards and are usually used for gaming.

With GPUs, mining became thousands of time more efficient than CPU mining. At this point, anyone mining on a CPU was providing such a tiny fraction of the hash rate that they were quickly unprofitable as the difficulty rose due to all the new GPU miners.

As GPUs took over and people started buying tons of graphics cards, efficiency of mining was tweaked even more through the production of ASICs (Application Specific Integrated Circuits). These are hardware computer chips that do only one thing - the bitcoin sha256 function and nothing else. Being specialized to this particular algorithm, ASICs were able to be thousands of times more efficient than GPUs for mining and quickly made GPUs unprofitable, just like GPUs had done to CPUs. Every few years, new generation of ASIC devices would put their earlier versions out of business with large efficiency improvements.

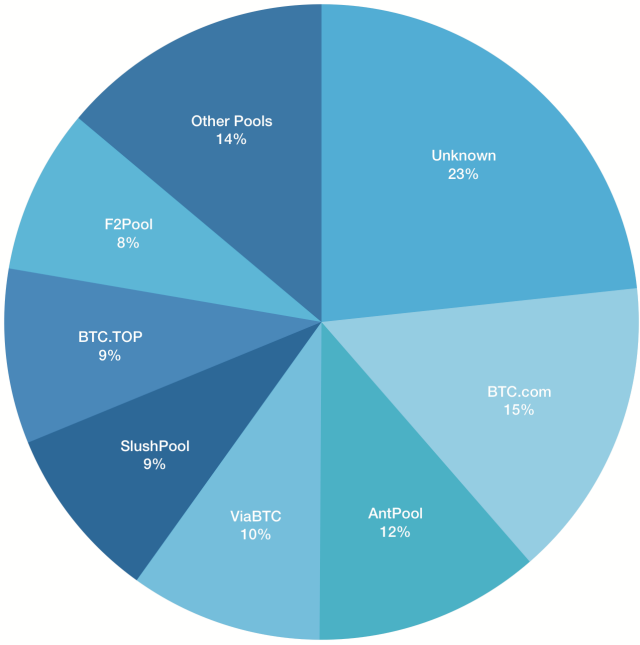
The first few miners on the network expended only a few pennies of electricity in order to produce their bitcoins. As the price of bitcoin rose, and more and more miners joined, the difficulty went up, and it became more and more expensive to generate bitcoins.

One issue with bitcoin mining is that it is nondeterministic, like rolling a die. That means you could end up spending hundreds of dollars in electricity and yet never find a valid block.

In 2010, an innovation called a *mining pool* (now known as Slushpool) emerged to address the problem of miners burning energy without receiving reward. A mining pool is a shared risk pool, similar to how medical insurance works.

All the miners contribute to mining for the pool, thus creating the appearance of one large miner. If anyone in the pool finds a valid block, the reward for the block is proportionally split amongst all the miners based on the hash rate they contributed. This allows even small mining operations such as individuals to receive reward for the small amount of hash rate they contribute. For providing this coordination service, the pool takes a cut of the rewards.

Mining pools caused an effect of centralization - users flocked to bigger pools. The diagram below shows the approximate mining pool distribution as of January 2019.



51% Attacks

Mining pool centralization leads to the worry that they could collude to 51% attack the network. If you look at the chart above, you'll note that the top 5 identifiable pools together have more than 50% of the total mining hash rate.

Let's examine how such an attack is performed, and what dangers it carries.

When you own just over 50% of the hash rate, you can dominate the writes into the ledger because you can produce a longer chain than the other less than 50% hash rate combined over time. Remember that Nakamoto Consensus says that nodes must accept the longest cumulative Proof of Work chain that they hear about.

Here's an example of how a very simple 51% attack is carried out:

1. Let's say the network as a whole is producing 1000 hashes/second and writing the blockchain.
2. You buy up a bunch of mining hardware and electricity to produce 2000 hashes/second. You now have 66% of the total hash rate (2000/3000).
3. You start mining a chain that contains only empty blocks.
4. Two weeks from now, you broadcast your chain of empty blocks. Because you are mining approximately twice as fast as the honest miners, your chain will be twice as long by cumulative Proof of Work. Broadcasting to all the existing nodes will cause them to reorg and lose the last two weeks of history.

Besides mining empty blocks, which makes the chain unusable, you can also perform a double spend attack:

1. Send some bitcoin to an exchange.
2. Exchange it to USD and withdraw the USD.
3. At some later date, broadcast a chain you mined secretly which does not contain the send to the exchange.
4. You've rewritten history and now have both the original bitcoin and the USD.

In practice, with Bitcoin's hash rate today (remember, it's using about the same amount of energy as a decent sized country), acquiring enough hardware and electricity to perform such an attack is extremely expensive. It's also very difficult to get away with a double-spend attack of this proportion without leaving footprints behind that could be used to figure out who you are. After all, you would be burning the energy of a medium sized country and buying up millions of dollars in hardware, and sending millions of dollars to exchanges in order to execute it.

Sustaining this kind of attack for any reasonable amount of time is infeasible, and if some malicious entity with unlimited funding did

decide to do this and was able to sustain this attack beyond the level of a nuisance, the network could adapt by changing to a different Proof of Work function (not sha256), which would render all the hardware ASICs used by the attacker completely useless. This, however, is the nuclear option as it would immediately put out of business all the honest miners as well. Nonetheless, the network would survive and arise from the ashes.

In addition to the infeasibility of the attack, having the majority of the hash rate does not entitle you to any of the following:

1. You can't create coins out of thin air. This violates the block reward consensus rule and your blocks would be rejected, even if they had enough Proof of Work.
2. You can't spend coins that aren't yours. You wouldn't be able to provide a valid digital signature, which violates the rules.
3. You can't cause a speed up the issuance schedule of Bitcoin. The difficulty would adjust every 2016 blocks as it always does.

Thus, the nodes that accept Bitcoin as payment would keep the network honest even in the face of a dishonest majority of miners. Furthermore, we should not assume that just because a mining pool has a specific percentage of hash rate that they own this hardware. In fact, most mining pools are comprised of thousands of individual miners. If the mining pool started misbehaving, these miners would have an incentive to switch away from the pool because they would want to protect the economic value of Bitcoin, which they are mining presumably to make money and not to lose it!

In fact, there is historic precedent for individual miners leaving a pool that became too powerful: In 2014, Ghash.io had close to half of the total mining power. Miners saw it creeping toward being too centralized and left for other pools voluntarily.

While relatively centralized mining pools are the reality today, there are

constant improvements to mining technology including a proposal called BetterHash, which lets individual miners be more in control of what they're mining and reduce reliance on coordination from pools.

Hard Forks and Soft Forks

We left the most complex topic in Bitcoin for last.

Hopefully by now you've got a good handle on how the Bitcoin software enforces the rules that people have agreed to, and how people can decide which software to run in order to enforce the rules that they believe in.

We've also covered that miners decide the rules they will follow when producing blocks, and that they must mine the kind of blocks that users want, or risk their blocks not being accepted and thus lose the mining reward.

Finally, we know that the Bitcoin software will accept the longest valid cumulative proof of work chain as the valid chain, and that forks sometimes occur naturally due to miners mining on out-of-date chain tips.

Now let's talk about intentional forks. An intentional fork is when some users and/or miners decide that they don't agree with the current rules of Bitcoin, and that they need to change the rules. There are two types of rule-changing forks that have been shown in the wild: soft-forks, which are backwards compatible, and hard-forks, which are not backwards compatible. Let's go through how these occur in theory, and then look at a few historical examples.

Soft Forks

A soft-fork is a backwards compatible change in the consensus rules of Bitcoin. What does that mean? It means that if you run an old node that has not upgraded to the new rules, your node will still see the

blocks produced under the new rules as valid. For a node upgraded with the new soft forked software, all blocks that were previously invalid remain invalid, but some valid blocks are now considered invalid. Let's look at an example to make it clear:

On Sep 12, 2010, a new rule was introduced to the software: blocks must be at most 1MB in size. This rule was introduced to deal with spam in the blockchain. Prior to this rule, all blocks of any size were valid. With the new rule, only smaller blocks were valid. If you were running an old node and didn't upgrade, then the new smaller blocks were still valid under your rules, so you were not affected.

A soft-fork is a non-disruptive way to upgrade the system because it allows node operators to upgrade to the new software slowly over time, voluntarily. If they don't upgrade, they'll still be able to process all the blocks coming in as they always did. Only the miners that produce the blocks have to upgrade to start producing blocks using the new rules. Once miners upgraded to the 1mb fork then all blocks from that point on were a max 1MB in size. Users running old versions of the software were none the wiser.

Hard Forks

A hard fork is the opposite of a soft fork. In the case of a hard-fork, a non-backwards-compatible change is introduced in which blocks that were originally invalid are now considered valid. In the case of a hard-fork, old nodes that did not upgrade will not be able to process the blocks produced under the new rules. Thus they will be stuck on the old chain unless they upgrade. An example of a hard fork would be one that changed the 1MB block size to something larger, as the blocks would be invalid under the old rules.

Majority hard forks that have near unanimous agreement from every node in the network would not cause problems. Every node would upgrade immediately to the new rules. If some stragglers were left

behind, they would not get any new block updates and would ideally notice that their software stopped working and be forced to upgrade.

In practice, hard forks never go quite so smoothly. In a truly decentralized anarchic system, you cannot coerce everyone to change to new rules. In August 2017, some people who were not happy with how the Bitcoin chain was progressing with regard to cheap payments decided that they wanted to fork to create a chain with larger blocks. Since Bitcoin had a rule about blocks being no more than 1MB (due to a soft-fork that had occurred in 2010), these people wanted to create a new chain with larger blocks. This fork became known as Bitcoin Cash.

An out-of-consensus hard fork like Bitcoin Cash, which is not followed by all miners and nodes, creates a new blockchain that shares some history with the original chain, but from the split point onwards, coins created on the fork are no longer Bitcoin as they are not accepted by any nodes on the Bitcoin network.

The subject what "is" or "is not" Bitcoin was hotly debated in the year following the Bitcoin Cash fork. There were some people on the Bitcoin Cash side who pushed a narrative that Bitcoin should be defined by what's written in the original design paper Satoshi produced ten years ago and cherry-picked specific words from the design paper to prove their point. But a consensus based system doesn't work based on arguments made on social media. It works by people choosing to run specific software to enforce specific rules.

In the case of this fork, the people running the vast majority of economically significant nodes - that is wallets, exchanges, and merchants did not want to change their software for something supported by a much smaller and less experienced development team and a much smaller amount of hash rate that had signaled that they wanted to change to these rules. Nor did people feel that such an "upgrade" was worth the disruption to the ecosystem. The problem with hard forks is that they only succeed when everyone switches. If

there are stragglers, two coins are created. Thus, Bitcoin remained Bitcoin, and Bitcoin Cash became a separate coin.

Today, dozens of other forks of Bitcoin exist, such as Bitcoin Gold, Bitcoin Diamond, and Bitcoin Private, with a tiny bit of hash rate securing them, low developer support, and nearly nonexistent economic activity. Many are outright scams or poorly thought out research projects. Hundreds more Bitcoin-like coins use similar code but do not share Bitcoin's account balance history (UTXO set), such as Litecoin or Dogecoin.

The Fee Market

We briefly mentioned transaction fees in Chapter 5 when discussing mining, but they deserve their own section. Since the Bitcoin issuance schedule consists of Block Reward Halvings every four years, until the Block Reward is entirely eliminated and Bitcoin enters into a state of zero additional supply until the end of time, we still need a way to incentivize miners to continue securing the network.

Fees are determined by a free market system where users bid for scarce space in a block. Users who send transactions indicate how much fee they are willing to pay to the miners, and miners may or may not include transactions that they see depending on the fees. When there are few transactions waiting to go into the next block, fees tend to be very low as there is no competition. As block space fills up, users are willing to pay higher fees for their transactions to be confirmed more quickly (within the next block). Those that don't want to pay can always set their fees low and wait longer to be mined at a later time when block space is more readily available.

Unlike in traditional financial systems, where fees tend to be based on a percentage of the amount being transferred, in Bitcoin the value being transferred has no bearing on the fees. Instead, we make fees proportional to the scarce resource they consume: block space. So fees are

measured in satoshis per byte (bytes are 8 bits, basically just a measure of how much data is in your transaction). Thus, a transaction that sends a million bitcoin from one address to another could actually be cheaper than one that consolidates 1 bitcoin scattered across 10 accounts because the latter requires more block space to represent.

In the past, there have been periods of time where Bitcoin was in very high demand, such as the massive bull run of late 2017, where fees became extremely high. Since that time, a few new features have been implemented to reduce fee pressure on the network.

One of them is called Segregated Witness, which reorganized how block data is represented by separating out the digital signatures from the transaction data and creating more space for this data - transactions that take advantage of this upgrade may use more than the original 1MB of block space through some clever tricks that are beyond the scope of this book.

The other relief to fees has come through batching: exchanges and other high volume players in the ecosystem started combining bitcoin transactions for multiple users into one transaction. Unlike a traditional payment in your bank or PayPal which is from one person to another, recall that a Bitcoin transaction can combine a large number of inputs and produce a large number of outputs. Thus, an exchange that needs to send bitcoin for withdrawal to 100 people can do so in a single transaction. This is a much more efficient use of block space, turning what is ostensibly only a handful of bitcoin transactions per second into thousands of payments per second.

Segregated Witness and batching have already done a very good job in reducing demand for block space. Further improvements are in the pipeline that make use of the block space more efficient. Nonetheless, there will come a time when Bitcoin fees become high again as blocks get more and more full due to demand.

Future Developments in Bitcoin

At this point, we've gone through inventing the protocol, and we've covered how the network evolved over time. Now we look to the future and cover some of the near term improvements coming to Bitcoin.

Unlike a traditional currency, which is a thing that is printed and used, Bitcoin is a programmable money layer on top of which we can build lots of services. This is an entirely new concept, and we're only starting to scratch the surface of what's possible.

Lightning Network

As we discussed above, Bitcoin has had issues with high fees as block space became more and more in demand. Today, Bitcoin is only capable of about 3 to 7 transactions per second based on the number of transactions that can fit into a block; remember, however, that each transaction can actually be a payment to hundreds of people via batching. Still, it is not enough capacity to become a global payment network.

A naive solution might be to raise the block size, and indeed several competing currencies including Bitcoin Cash have tried this approach. Bitcoin does not go this route because increasing the block size would negatively impact decentralization characteristics such as the number of nodes and how geographically dispersed they are. Even if a block size increase was possible due to improvements in hardware, there is also the issue that Bitcoin's decentralized nature means that a hard fork that tries to change the block size would cause a lot of disruption, and likely another outright split into a different coin.

A block size increase would also not really solve the problem of making Bitcoin suitable as a worldwide payment system—it simply wouldn't scale that much. Enter the Lightning Network: another protocol and set of software implementations that create off-chain Bitcoin transactions.

The Lightning Network could be the topic of an entire book, but we'll discuss it briefly.

The idea of Lightning is that not every transaction needs to be recorded to the blockchain. For example, if you and I are at a bar buying drinks, we can open a bar tab and settle at the end of the night. It doesn't really make sense for us to charge our credit card for every drink as it wastes time. With Bitcoin, using the energy equivalent to that of an entire country on confirming the purchase of a coffee or beer and having this purchase recorded for all of time on thousands of computers across the world is neither scalable nor particularly good for privacy.

The Lightning Network, if successful, will improve on many of Bitcoin's downsides:

- Virtually unlimited transaction throughput. Hundreds of thousands of micro transactions could be performed and committed to the Bitcoin blockchain once as final settlement.
- Instant confirmations; no need to wait for blocks to be mined.
- Sub-penny transaction fees suitable for micropayments such as paying a penny to read a blog.
- Increased privacy. Only the parties participating in the transaction need to know about it, unlike an on-chain transaction which is broadcast to the entire world.

Lightning uses the concept of Payment Channels, which are real on-chain Bitcoin transactions that lock up some amount of Bitcoin and make it available within the Lightning Network for instant, near-free transfer. The Lightning Network is in early stages but already shows promise. You can check out a site that uses Lightning-based micropayments for articles at https://yalls.org/.

Bitcoin in Space

Bitcoin does an excellent job of being censorship resistant as it is resistant to seizure (you can carry it in your head), and resistant to censor-

ship of transfer since it only requires one honest miner on the network to commit your transactions (and you can mine yourself).

Nonetheless, being that Bitcoin is transmitted over the Internet, it is susceptible to censorship on the network level. Authoritarian regimes that want to clamp down on activity could attempt to block Bitcoin traffic entering and leaving their country.

The Blockstream Satellite is the first effort to route around state-level network censorship, as well as reach remote areas that may not have connections to the Internet. This satellite allows anyone with a dish and relatively inexpensive equipment to connect and download the Bitcoin blockchain, with bi-directional communication coming soon. There are also now efforts such as TxTenna to build off-the-grid mesh networks. When coupled with a satellite connection, this kind of setup would be nearly unstoppable.

10

WHAT'S NEXT?

So this is it. You've gone through the exercise of Inventing Bitcoin, and hopefully emerged on the other side of the looking glass, ready to explore further. Where do you go from here? Here are a few resources to help you explore further:

To learn more about the economics behind Bitcoin:

- The Bitcoin Standard by Saifedean Ammous
- Cryptoassets by Chris Burniske and Jack Tatar
- Google: Austrian Economics
- Bitcoin Investment Theses by Pierre Rochard https://medium.com/@pierre_rochard/bitcoin-investment-theses-part-1-e97670b5389b
- The Bullish Case for Bitcoin by Vijay Boyapati https://medium.com/@vijayboyapati/the-bullish-case-for-bitcoin-6ecc8bdecc1

To get deeper on the computer science:

- The Bitcoin whitepaper by Satoshi

https://bitcoin.org/bitcoin.pdf
- Mastering Bitcoin by Andreas Antonopulous
- Jimmy Song's seminar at https://programmingblockchain.com/ and his book on github at https://programmingblockchain.gitbook.io/programmingblockchain

To get deeper on the history and philosophy of Bitcoin:

- Planting Bitcoin by Dan Held https://medium.com/@danhedl/planting-bitcoin-sound-money-72e80e40ff62,
- Bitcoin Governance by Pierre Richard https://medium.com/@pierre_rochard/bitcoin-governance-37e86299470f
- Bitcoin Past and Future by Murad Mahmudov https://blog.usejournal.com/bitcoin-past-and-future-45d92b3180f1
- Every video made by Andreas Antonopulous, especially Currency Wars and The Monument of Immutability, at https://www.youtube.com/user/aantonop

A huge part of the Bitcoin ecosystem lives on Twitter. Here's a handful of folks in no particular order that are good to follow. Start here, and branch out:

```
@lopp
@pwuille
@adam3us
@danheld
@TraceMayer
@pierre_rochard
@bitstein
@Melt_Dem
```

```
@theonevortex
@WhatBitcoinDid
@stephanlivera
@TheBlock__
@TheLTBNetwork
@real_vijay
@jimmysong
@Excellion
@starkness
@roasbeef
@saifedean
@giacomozucco
@Snyke
@aantonop
@MustStopMurad
@peterktodd
@skwp (that's me)
```

You can find more of my writing at yanpritzker.com. See you on the other side.

ACKNOWLEDGMENTS

Thank you to the many people who gave me feedback during early drafts of this book. In particular: Joe Levering, Phil Geiger, Yury Pritzker, Jonathan Wheeler, and Walter Rosenberg.

Thank you to Jimmy Song for your Programming Blockchain seminar, which gave me the swift kick in the ass I needed to put this text together.

ABOUT THE AUTHOR

Yan Pritzker has been a developer and startup entrepreneur for the last 20 years. Most recently, he was the co-founding CTO at Reverb.com where he ran technology and infrastructure from 2012-2018. Today he is focused on Bitcoin education and consulting for early stage startups.

Yan writes on Bitcoin and related topics at yanpritzker.com.

You can also follow him on Twitter: @skwp.