

Bitcoin and Cryptocurrency Technologies

**Arvind Narayanan, Joseph Bonneau, Edward Felten,
Andrew Miller, Steven Goldfeder**

Draft — Oct 6, 2015

Feedback welcome! Email bitcoinbook@lists.cs.princeton.edu

Chapter 4: How to Store and Use Bitcoins

This chapter is about how we store and use bitcoins in practice.

4.1 Simple Local Storage

Let's begin with the simplest way of storing bitcoins, and that is simply putting them on a local device. As a recap, to spend a bitcoin you need to know some public information and some secret information. The public information is what goes on the block chain — the identity of the coin, how much it's worth, and so on. The secret information is the secret key of the owner of the bitcoin, presumably, that's you. You don't need to worry too much about how to store the public information because you can always get it back when you need to. But the secret signing key is something you'd better keep track of. So in practice storing your bitcoins is all about storing and managing your keys.

Storing bitcoins is really all about storing and managing Bitcoin secret keys.

When figuring out how to store and manage keys, there are three goals to keep in mind. The first is availability: being able to actually spend your coins when you want to. The second is security: making sure that nobody else can spend your coins. If someone gets the power to spend your coins they could just send your coins to themselves, and then you don't have the coins anymore. The third goal is convenience, that is, key management should be relatively easy to do. As you can imagine, achieving all three simultaneously can be a challenge.

Different approaches to key management offer different trade-offs between availability, security, and convenience.

The simplest key management method is storing them on a file on your own local device: your computer, your phone, or some other kind of gadget that you carry, or own, or control. This is great for convenience: having a smartphone app that allows spending coins with the push of a few buttons is hard to beat. But this isn't great for availability or security — if you lose the device, if the device crashes, and you have to wipe the disc, or if your file gets corrupted, your keys are lost, and so are your coins. Similarly for security: if someone steals or breaks into your device, or it gets infected with malware, they can copy your keys and then they can then send all your coins to themselves.

In other words, storing your private keys on a local device, especially a mobile device, is a lot like carrying around money in your wallet or in your purse. It's useful to have some spending money, but you don't want to carry around your life savings because you might lose it, or somebody might steal it. So what you typically do is store a little bit of information/a little bit of money in your wallet, and keep most of your money somewhere else.

Wallets. If you're storing our bitcoins locally, you'd typically use wallet software, which is software that keeps track of all your coins, manages all the details of your keys, and makes things convenient with a nice user interface. If you want to send \$4.25 worth of bitcoins to your local coffee shop the wallet software would give you some easy way to do that. Wallet software is especially useful because you typically want to use a whole bunch of different addresses with different keys associated with them. As you may remember, creating a new public/private key pair is easy, and you can utilize this to improve your anonymity or privacy. Wallet software gives you a simple interface that tells you how much is in your wallet. When you want to spend bitcoins, it handles the details of which keys to use and how to generate new addresses and so on.

Encoding keys: base 58 and QR codes. To spend or receive bitcoins, you also need a way to exchange an address with the other party — the address to which bitcoins are to be sent. There are two main ways in which addresses are encoded so that they can be communicated from receiver to spender: as a text string or as a QR code.

To encode an address as a text string, we take the bits of the key and convert it from a binary number to a base 58 number. Then we use a set of 58 characters to encode each digit as a character; this is called base58 notation. Why 58? Because that's the number we get when we include the upper case letters, lower case letters, as well as digits as characters, but leave out a few that might be confusing or might look like another character. For example, capital letter 'O' and zero are both taken out because they look too much alike. This allows encoded addresses to be read out over the phone or read from printed paper and typed in, should that be necessary. Ideally such manual methods of communicating addresses can be avoided through methods such as QR codes, which we now discuss.

1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa

The address that received the very first Bitcoin block reward in the genesis block, base58 encoded.



Figure 4.1: a QR code representing an actual Bitcoin address. Feel free to send us some bitcoins.

The second method for encoding a Bitcoin address is as a QR code, a simple kind of 2-dimensional barcode. The advantage of a QR code is that you can take a picture of it with a smartphone and wallet

software can automatically turn the barcode into the a sequence of bits that represents the corresponding Bitcoin address. This is useful in a store, for example: the check-out system might display a QR code and you can pay with your phone by scanning the code and sending coins to that address. It is also useful for phone-to-phone transfers.

4.2 Hot and Cold Storage

As we just saw, storing bitcoins on your computer is like carrying money around in your wallet or your purse. This is called “hot storage”. It’s convenient but also somewhat risky. On the other hand, “cold storage” is offline. It's locked away somewhere. It's not connected to the internet, and it's archival. So it’s safer and more secure, but of course, not as convenient. This is similar to how you carry some money around on your person, but put your life's savings somewhere safer.

To have separate hot and cold storage, obviously you need to have separate secret keys for each — otherwise the coins in cold storage would be vulnerable if the hot storage is compromised. You’ll want to move coins back and forth between the hot side and the cold side, so each side will need to know the other’s addresses, or public keys.

Cold storage is not online, and so the hot storage and the cold storage won't be able to connect to each other across any network. But the good news is that cold storage doesn’t have to be online to receive coins — since the hot storage knows the cold storage addresses, it can send coins to cold storage at any time. At any time if the amount of money in your hot wallet becomes uncomfortably large, you can transfer a chunk of it over to cold storage, without putting your cold storage at risk by connecting to the network. Next time the cold storage connects it will be able to receive from the block chain information about those transfers to it and then the cold storage will be able to do what it wants with those coins.

But there’s a little problem when it comes to managing cold storage addresses. On the one hand, as we saw earlier, for privacy and other reasons we want to be able to receive each coin at a separate address with different secret keys. So whenever we transfer a coin from the hot side to the cold side we'd like to use a fresh cold address for that purpose. But because the cold side is not online we have to have some way for the hot side to find out about those addresses.

The blunt solution is for the cold side generate a big batch of addresses all at once and send those over for the hot side to use them up one by one. The drawback is that we have to periodically reconnect the cold side in order to transfer more addresses.

Hierarchical wallets. A more effective solution is to use a hierarchical wallet. It allows the cold side to use an essentially unbounded number of addresses and the hot side to know about these addresses, but with only a short, one-time communication between the two sides. But it requires a little bit of cryptographic trickery.

To review, previously when we talked about key generation and digital signatures back in chapter 1, we looked at a function called `generateKeys` that generates a public key (which acts as an address) and a secret key. In a hierarchical wallet, key generation works differently. Instead of generating a single address we generate what we'll call address generation info, and rather than a private key we generate what we'll call private key generation info. Given the address generation info, we can generate a sequence of addresses: we apply an address generation function that takes as input the address generation info and any integer i and generate the i 'th address in the sequence. Similarly we can generate a sequence of private keys using the private key generation info.

The cryptographic magic that makes this useful is that for every i , the i 'th address and i 'th secret key "match up" — that is, the i 'th secret key controls, and can be used to spend, bitcoins from the i 'th address just as if the pair were generated the old fashioned way. So it's as if we have a sequence of regular key pairs.

The other important cryptographic property here is security: the address generation info doesn't leak any information about the private keys. That means that it's safe to give the address generation info to anybody, and so that anybody can be enabled to generate the i 'th key.

Now, not all digital signature schemes that exist can be modified to support hierarchical key generation. Some can and some can't, but the good news is that the digital signature scheme used by Bitcoin, ECDSA, does support hierarchical key generation, allowing this trick. That is, the cold side generates an arbitrarily many keys and the hot side generates the corresponding addresses.

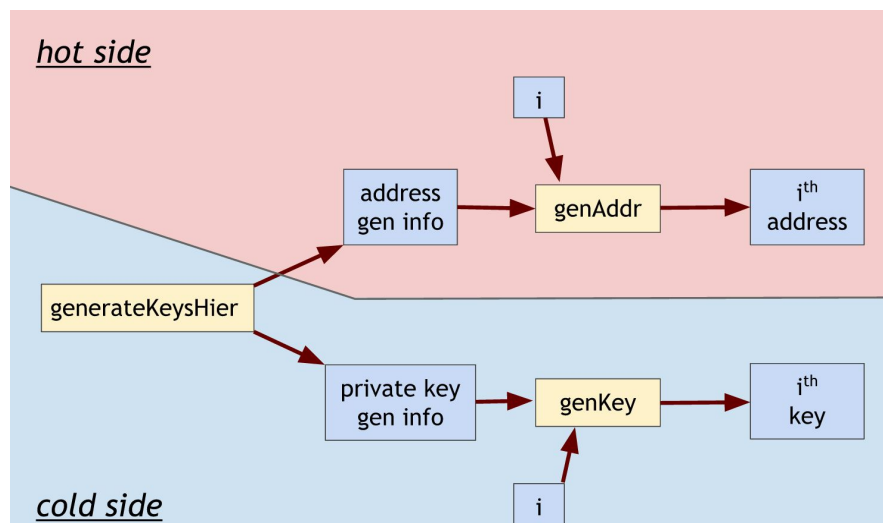


Figure 4.2: Schema of a hierarchical wallet. The cold side creates and saves private key generation info and address generation info. It does a one-time transfer of the latter to the hot side. The hot side generates a new address sequentially every time it wants to send coins to the cold side. When the cold side reconnects, it generates addresses sequentially and checks the block chain for transfers to those addresses until it reaches an address that hasn't received any coins. It can also generate private keys sequentially if it wants to send some coins back to the hot side or spend them some other way.

Now let's talk about the different ways in which cold information — whether one or more keys, or key-generation info — can be stored. The first way is to store it in some kind of device and put that device in a safe. It might be a laptop computer, a mobile phone or tablet, or a thumb drive. The important thing is to turn the device off and lock it up, so that if somebody wants to steal it they have to break into the locked storage.

Brain wallet. The second method we can use is called a brain wallet. This is a way to control access to bitcoins using nothing but a secret passphrase. This avoids the need for hard drives, paper, or any other long-term storage mechanism. This property can be particularly useful in situations where you have poor physical security, perhaps when you're traveling internationally.

The key trick behind a brain wallet is to have a predictable algorithm for turning a passphrase into a public and private key. For example, you could hash the passphrase with a suitable hash function to derive the private key, and given the private key, the public key can be derived in a standard way. Further, combining this with the hierarchical wallet technique we saw earlier, we can generate an entire sequence of addresses and private keys from a passphrase, thus enabling a complete wallet.

However, an adversary can also obtain all private keys in a brain wallet if they can guess the passphrase. As always in computer security, we must assume that the adversary knows the procedure you used to generate keys, and only your passphrase provides security. So the adversary can try various passphrases and generate addresses using them; if he finds any unspent transactions on the block chain at any of those addresses, he can immediately transfer them to himself. The adversary may never know (or care) who the coins belonged to and the attack doesn't require breaking into any machines. Guessing brain wallet passphrases is not directed toward specific users, and further, leaves no trace.

Furthermore, unlike the task of guessing your email password which can be *rate-limited* by your email server (called *online guessing*), with brain wallets the attacker can download the list of addresses with unredeemed coins and try as many potential passphrases as they have the computational capacity to check. Note that the attacker doesn't need to know which addresses correspond to brain wallets. This is called *offline guessing* or *password cracking*. It is much more challenging to come up with passphrases that are easy to memorize and yet won't be vulnerable to guessing in this manner. One secure way to generate a passphrase is to have an automatic procedure for picking a random 80-bit number and turning that number into a passphrase in such a way that different numbers result in different passphrases.

In practice, it is also wise to use a deliberately slow function to derive the private key from the passphrase (referred to as *key stretching*) to ensure it takes as long as possible for the attacker to try all possibilities. The basic approach is to take a fast cryptographic hash function like SHA-256 and compute perhaps 2^{20} iterations of it, multiplying the attacker's workload by a factor of 2^{20} . Of course, if it is too slow it will start to become annoying to the user as their device must re-compute this function any time they want to spend coins from their brain wallet.

If a brain wallet passphrase is inaccessible — say it's been forgotten, hasn't been written down, and can't be guessed — then the coins are lost forever.

Paper wallet. The third option is what's called a paper wallet. We can print the key material to paper and then put that paper into a safe or secure place. Obviously, the security of this method is just as good or bad as the physical security of the paper that we're using. Typical paper wallets encode both the public and private key in two ways: as a 2D barcode and in base 58 notation. Just like with a brain wallet, storing a small amount of key material is sufficient to re-create a wallet.

Tamper-resistant device. The fourth way that we can store offline information is to put it in some kind of tamper-resistant device. Either we put the key into the device or the device generates the key; either way, the device is designed so that there's no way it will output or divulge the key. The device instead signs statements with the key, and does so when we, say, press a button or give it some kind of password. One advantage is that if the device is lost or stolen we'll know it, and the only way the key can be stolen is if the device is stolen. This is different from storing your key on a laptop.

In general, people might use a combination of four of these methods in order to secure their keys. For hot storage, and especially for hot storage holding large amounts of bitcoins, people are willing to work pretty hard and come up with novel security schemes in order to protect them, and we'll talk a little bit about one of those more advanced schemes in the next section.

4.3 Splitting and Sharing Keys

Up to now we've looked at different ways of storing and managing the secret keys that control bitcoins, but we've always put a key in a single place — whether locked in a safe, or in software, or on paper. This leaves us with a single point of failure. If something goes wrong with that single storage place then we're in trouble. We could create and store backups of the key material, but while this decreases the risk of the key getting lost or corrupted (availability), it *increases* the risk of theft (security). This trade-off seems fundamental. Can we take a piece of data and store it in such a way that availability and security increase at the same time? Remarkably, the answer is yes, and it is once again a trick that uses cryptography, called *secret sharing*.

Here's the idea: we want to divide our secret key into some number N of pieces. We want to do it in such a way that if we're given any K of those pieces then we'll be able to reconstruct the original secret, but if we're given fewer than K pieces then we won't be able to learn anything about the original secret.

Given this stringent requirement, simply "cutting up" the secret into pieces won't work because even a single piece gives some information about the secret. We need something cleverer. And since we're not cutting up the secret, we'll call the individual components "shares" instead of pieces.

Let's say we have $N=2$ and $K=2$. That means we're generating 2 shares based on the secret, and we need both shares to be able to reconstruct the secret. Let's call our secret S , which is just a big (say 128-bit) number. We could generate a 128-bit random number R and make the two shares be R and $S \oplus R$. (\oplus represents bitwise XOR). Essentially, we've "encrypted" S with a one-time pad, and we store the key (R) and the ciphertext ($S \oplus R$) in separate places. Neither the key nor the ciphertext by itself tells us anything about the secret. But given the two shares, we simply XOR them together to reconstruct the secret.

This trick works as long as N and K are the same — we'd just need to generate $N-1$ different random numbers for the first $N-1$ shares, and the final share would be the secret XOR'd with all other $N-1$ shares. But if N is more than K , this doesn't work any more, and we need some algebra.

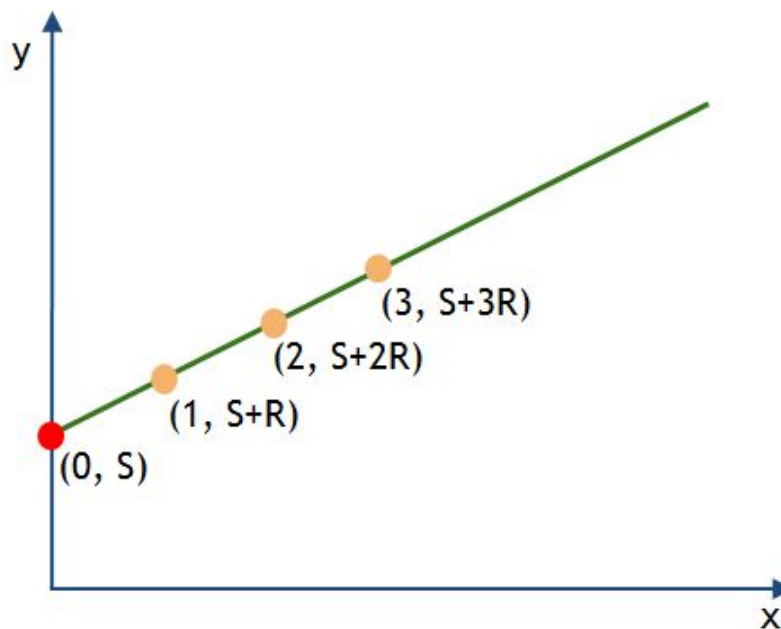


Figure 4.3: Geometric illustration of 2-out-of- N secret sharing. S represents the secret, encoded as a (large) integer. The green line has a slope chosen at random. The orange points (specifically, their Y -coordinates $S+R$, $S+2R$, ...) correspond to shares. Any two orange points are sufficient to reconstruct the red point, and hence the secret. All arithmetic is done modulo a large prime number.

Take a look at Figure 4.3. What we've done here is to first generate the point $(0, S)$ on the Y -axis, and then drawn a line with a random slope through that point. Next we generate a bunch of points on that line, as many as we want. It turns out that this is a secret sharing of S with N being the number of points we generated and $K=2$.

Why does this work? First, if you're given two of the points generated, you can draw a line through them and see where it meets the Y -axis. That would give you S . On the other hand, if you're given only a single point, it tells you nothing about S , because the slope of the line is random. Every line through your point is equally likely, and they would all intersect the Y -axis at different points.

There's only one other subtlety: to make the math work out, we have to do all our arithmetic modulo a large prime number P . It doesn't need to be secret or anything, just really big. And the secret S has to be between 0 and $P-1$, inclusive. So when we say we generate points on the line, what we mean is that we generate a random value R , also between 0 and $P-1$, and the points we generate are

$$x=1, y=(S+R) \bmod P$$

$$x=2, y=(S+2R) \bmod P$$

$$x=3, y=(S+3R) \bmod P$$

and so on. The secret corresponds to the point $x=0, y=(S+0 \cdot R) \bmod P$, which is just $x=0, y=S$.

What we've seen is a way to do secret sharing with $K=2$ and any value of N . This is already pretty good — if $N=4$, say, you can divide your secret key into 4 shares and put them on 4 different devices so that if someone steals any one of those devices, they learn nothing about your key. On the other hand, even if two of those devices are destroyed in a fire, you can reconstruct the key using the other two. So as promised, we've increased both availability and security.

But we can do better: we can do secret sharing with any N and K as long as K is no more than N . To see how, let's go back to the figure. The reason we used a line instead of some other shape is that a line, algebraically speaking, is a polynomial of degree 1. That means that to reconstruct a line we need two points and no fewer than two. If we wanted $K=3$, we would have used a parabola, which is a quadratic polynomial, or a polynomial of degree 2. Exactly three points are needed to construct a quadratic function. We can use the table below to understand what's going on.

Equation	Degree	Shape	Random parameters	Number of points (K) needed to recover S
$(S + RX) \bmod P$	1	Line	R	2
$(S + R_1X + R_2X^2) \bmod P$	2	Parabola	R_1, R_2	3
$(S + R_1X + R_2X^2 + R_3X^3) \bmod P$	3	Cubic	R_1, R_2, R_3	4

Table 4.1: The math behind secret sharing. Representing a secret via a series of points on a random polynomial curve of degree $K-1$ allows the secret to be reconstructed if, and only if, at least K of the points ("shares") are available.

There is a formula called Lagrange interpolation that allows you to reconstruct a polynomial of degree $K-1$ from any K points on its curve. It's an algebraic version (and a generalization) of the geometric intuition of drawing a straight line through two points with a ruler. As a result of all this, we have a way to store any secret as N shares such that we're safe even if an adversary learns up to $K-1$ of them, and at the same time we can tolerate the loss of up to $N-K$ of them.

None of this is specific to Bitcoin, by the way. You can secret-share your passwords right now and give shares to your friends or put them on different devices. But no one really does this with secrets like passwords. Convenience is one reason; another is that there are other security mechanisms available for important online accounts, such as two-factor security using SMS verification. But with Bitcoin, if you're storing your keys locally, you don't have those other security options. There's no way to make the control of a Bitcoin address dependent on receipt of an SMS message. The situation is different with online wallets, which we'll look at in the next section. But not too different — it just shifts the problem to a different place. After all, the online wallet provider will need some way to avoid a single point of failure when storing *their* keys.

Threshold cryptography. But there's still a problem with secret sharing: if we take a key and we split it up in this way and we then want to go back and use the key to sign something, we still need to bring the shares together and recalculate the initial secret in order to be able to sign with that key. The point where we bring all the shares together is still a single point of vulnerability where an adversary might be able to steal the key.

Cryptography can solve this problem as well: if the shares are stored in different devices, there's a way to produce Bitcoin signatures in a decentralized fashion without ever reconstructing the private key on any single device. This is called a "threshold signature." The best use-case is a wallet with two-factor security, which corresponds to the case $N=2$ and $K=2$. Say you've configured your wallet to split its key material between your desktop and your phone. Then you might initiate a payment on your desktop, which would create a partial signature and send it to your phone. Your phone would then alert you with the payment details — recipient, amount, etc. — and request your confirmation. If the details check out, you'd confirm, and your phone would complete the signature using its share of the private key and broadcast the transaction to the block chain. If there were malware on your desktop that tried to steal your bitcoins, it might initiate a transaction that sent the funds to the hacker's address, but then you'd get an alert on your phone for a transaction you didn't authorize, and you'd know something was up. The mathematical details behind threshold signatures are complex and we won't discuss them here.

Multi-signatures. There's an entirely different option for avoiding a single point of failure: multi-signatures, which we saw earlier in Chapter 3. Instead of taking a single key and splitting it, Bitcoin script directly allows you to stipulate that control over an address be split between different keys. These keys can then be stored in different locations and the signatures produced separately. Of course, the completed, signed transaction will be constructed on some device, but even if the adversary controls this device, all that he can do is to prevent it from being broadcast to the network. He can't produce valid multi-signatures of some other transaction without the involvement of the other devices.

As an example, suppose that Andrew, Arvind, Ed, Joseph, and Steven, the authors of this book, are co-founders of a company — perhaps we started it with the copious royalties from the sale of this free book — and the company has a lot of bitcoins. We might use multi-sig to protect our large store

of bitcoins. Each of the five of us will generate a key pair, and we'll protect our cold storage using 3-out-of-5 multi-sig, which means that three of us must sign to create a valid transaction.

As a result, we know that we're relatively secure if the five of us keep our keys separately and secure them differently. An adversary would have to compromise three out of the five keys. If one or even two of us go rogue, they can't steal the company's coins because you need at least three keys to do that. At the same time, if one of us loses our key or gets run over by a bus and our brain wallet is lost, the others can still get the coins back and transfer them over to a new address and re-secure the keys. In other words, multi-sig helps you to manage large amounts of cold-stored coins in a way that's relatively secure and requires action by multiple people before anything drastic happens.

Sidebar. Threshold signatures are a cryptographic technique to take a single key, split it into shares, store them separately, and sign transactions without reconstructing the key. Multi-signatures are a feature of Bitcoin script by which you can specify that control of an address is split between multiple independent keys. While there are some differences between them, they both increase security by avoiding single points of failure.

4.4 Online Wallets and Exchanges

So far we've talked about ways in which you can store and manage your bitcoins yourself. Now we'll talk about ways you can use other people's services to help you do that. The first thing you could do is use an online wallet.

Online wallets. An online wallet is kind of like a local wallet that you might manage yourself, except the information is stored in the cloud, and you access it using a web interface on your computer or using an app on your smartphone. Some online wallet services that are popular in early 2015 are Coinbase and blockchain.info.

What's crucial from the point of view of security is that the site delivers the code that runs on your browser or the app, and it also stores your keys. At least it will have the ability to access your keys. Ideally, the site will encrypt those keys under a password that only you know, but of course you have to trust them to do that. You have to trust their code to not leak your keys or your password.

An online wallet has certain trade offs to doing things yourself. A big advantage is that it's convenient. You don't have to install anything on your computer in order to be able to use an online wallet in your browser. On your phone you maybe just have to install an app once, and it won't need to download the block chain. It will work across multiple devices — you can have a single wallet that you access on your desktop and on your phone and it will just work because the real wallet lives in the cloud.

On the other hand, there are security worries. If the site or the people who operate the site turn out to be malicious or are compromised somehow, your bitcoins are in trouble. The site supplies the code that has its grubby fingers on your bitcoins, and things can go wrong if there's a compromise or a malice at the service provider.

Ideally, the site or the service is run by security professionals who are better trained, or perhaps more diligent than you in maintaining security. So you might hope that they do a better job and that your coins are actually more secure than if you stored them yourself. But at the end of day, you have to trust them and you have to rely on them not being compromised.

Bitcoin exchanges. To understand Bitcoin exchanges, let's first talk about how banks or bank like services operate in the traditional economy. You give the bank some money — a deposit — and the bank promises to give you back that money later. Of course, crucially, the bank doesn't actually just take your money and put it in a box in the back room. All the bank does is promise that if you show up for the money they'll give it back. The bank will typically take the money and put it somewhere else, that is, invest it. The bank will probably keep some money around in reserve in order to make sure that they can pay out the demand for withdrawals that they'll face on a typical day, or maybe even an unusual day. Many banks typically use something called ***fractional reserve*** where they keep a certain fraction of all the demand deposits on reserve just in case.

Now, Bitcoin exchanges are businesses that at least from the user interface standpoint function in a similar way to banks. They accept deposits of bitcoins and will, just like a bank, promise to give them back on demand later. You can also transfer fiat currency — traditional currency like dollars and euros — into an exchange by doing a transfer from your bank account. The exchange promises to pay back either or both types of currency on demand. The exchange lets you do various banking-like things. You can make and receive Bitcoin payments. That is, you can direct the exchange to pay out some bitcoins to a particular party, or you can ask someone else to deposit funds into the particular exchange on your behalf — put into your account. They also let you exchange bitcoins for fiat currency or vice versa. Typically they do this by finding some customer who wants to buy bitcoins with dollars and some other customer who wants to sell bitcoins for dollars, and match them up. In other words, they try to find customers willing to take opposite positions in a transaction. If there's a mutually acceptable price, they will consummate that transaction.

Suppose my account at some exchange holds 5000 dollars and three bitcoins and I use the exchange, I put in an order to buy 2 bitcoins for 580 dollars each, and the exchange finds someone who is willing to take the other side of that transaction and the transaction happens. Now I have five bitcoins in my account instead of three, and 3840 dollars instead of 5000.

The important thing to note here is that when this transaction happened involving me and another customer of the same exchange, no transaction actually happened on the Bitcoin block chain. The exchange doesn't need to go to the block chain in order to transfer bitcoins or dollars from account to another. All that happens in this transaction is that the exchange is now making a different promise to me than they were making before. Before they said, "we'll give you 5000 USD and 3 BTC" and now

they're saying "we'll give you 3840 USD and 5 BTC." It's just a change in their promise — no actual movement of money through the dollar economy or through the block chain. Of course, the other person has had their promises to them change in the opposite way.

There are pros and cons to using exchanges. One of the big pros is that exchanges help to connect the Bitcoin economy and the flows of bitcoins with the fiat currency economy so that it's easy to transfer value back and forth. If I have dollars and bitcoins in my account I can trade back and forth between them pretty easily, and that's really helpful.

The con is risk. You have the same kind of risk that you face with banks, and those risks fall into three categories.

Three types of risks. The first risk is the risk of a **bank run**. A run is what happens when a bunch of people show up all at once and want their money back. Since the bank maintains only fractional reserves, it might be unable to cope with the simultaneous withdrawals. The danger is a kind of panic behavior where once the rumor starts to get around that a bank or exchange might be in trouble and they might be getting close to not honoring withdrawals, then people stampede in to try to withdraw their money ahead of the crowd, and you get a kind of avalanche.

The second risk is that the owners of the banks might just be crooks running a Ponzi scheme. This is a scheme where someone gets people to give them money in exchange for profits in the future, but then actually takes their money and uses it to pay out the profits to people who bought previously. Such a scheme is doomed to eventually fail and lose a lot of people a lot of money. Bernie Madoff most famously pulled this off in recent memory.

The third risk is that of a hack, the risk that someone — perhaps even an employee of the exchange — will manage to penetrate the security of the exchange. Since exchanges store key information that controls large amounts of bitcoins, they need to be really careful about their software security and their procedures — how they manage their cold and hot storage and all of that. If something goes wrong, your money could get stolen from the exchange.

All of these things have happened. We have seen exchanges that failed due to the equivalent of a bank run. We've seen exchanges fail due to the operators of the exchange being crooks, and we've seen exchanges that fail due to break-ins. In fact, the statistics are not encouraging. A study in 2013 found that 18 of 40 Bitcoin exchanges had ended up closing due to some failure or some inability to pay out the money that the exchange had promised to pay out.

The most famous example of this of course is Mt. Gox. Mt. Gox was at one time the largest Bitcoin exchange, and it eventually found itself insolvent, unable to pay out the money that it owed. Mt. Gox was a Japanese company and it ended up declaring bankruptcy and leaving a lot of people wondering where their money had gone. Right now the bankruptcy of Mt. Gox is tangled up in the Japanese and American courts, and it's going to be a while before we know exactly where the money went. The one

thing we know is that there's a lot of it and Mt. Gox doesn't have it anymore. So this is a cautionary tale about the use of exchanges.

Connecting this back to banks, we don't see a 45% failure rate for banks in most developed countries, and that's partly due to regulation. Governments regulate traditional banks in various ways.

Bank regulation. The first thing that governments do is they often impose a minimum reserve requirement. In the U.S., the fraction of demand deposits that banks are required to have in liquid form is typically 3-10%, so that it can deal with a surge of withdrawals if that happens. Second, governments often regulate the types of investments and money management methods that banks can use. The goal is to ensure that the banks' assets are invested in places that are relatively low risk, because those are really the assets of the depositors in some sense.

Now, in exchange for these forms of regulation governments typically do things to help banks or help their depositors. First, governments will issue deposit insurance. That is, the government promises depositors that if a bank that follows these rules goes under, the government will make good on at least part of those deposits. Governments also sometimes act as a "lender of last resort." If a bank gets itself into a tough spot, but it's basically solvent, the government may step in and loan the bank money to tide it over until it can move money around as necessary to get itself out of the woods.

So, traditional banks are regulated in this way. Bitcoin exchanges are not. The question of whether or how Bitcoin exchanges or other Bitcoin business should be regulated is a topic that we will come back to in chapter 7.

Proof of reserve. A Bitcoin exchange or someone else who holds bitcoins can use a cryptographic trick called a proof of reserve to give customers some comfort about the money that they deposited. The goal is for the exchange or business holding bitcoins to prove that it has a fractional reserve — that they retain control of perhaps 25% or maybe even 100% of the deposits that people have made.

We can break the proof-of-reserve problem into two pieces. The first is to prove how much reserve you're holding — that's the relatively easy part. The company simply publishes a valid payment-to-self transaction of the claimed reserve amount. That is, if they claim to have 100,000 bitcoins, they create a transaction in which they pay 100,000 bitcoins to themselves and show that that transaction is valid. Then they sign a challenge string — a random string of bits generated by some impartial party — with the same private key that was used to sign the payment-to-self transaction. This proves that someone who knew that private key participated in the proof of reserve.

We should note two caveats. Strictly speaking, that's not a proof that the party that's claiming to own the reserve owns it, but only that whoever does own those 100,000 bitcoins is willing to cooperate in this process. Nonetheless, this looks like a proof that somebody controls or knows someone who controls the given amount of money. Also, note that you could always under-claim: the organization might have 150,000 bitcoins but choose to make a payment-to-self of only 100,000. So this proof of reserve doesn't prove that this all you have, but it proves that you have at least that much.

Proof of liabilities. The second piece is to prove how many demand deposits you hold, which is the hard part. If you can prove your reserves and you demand deposits then anyone can simply divide those two numbers and that's what your fractional reserve is. We'll present a scheme that allows you to *over-claim* but not under-claim your demand deposits. So if you can prove that your reserves are at least a certain amount and your liabilities are at most a certain amount, taken together, you've proved a lower bound on your fractional reserve.

If you didn't care at all about the privacy of your users, you could simply publish your records — specifically, the username and amount of every customer with a demand deposit. Now anyone can calculate your total liabilities, and if you omitted any customer or lied about the value of their deposit, you run the risk that that customer will expose you. You could make up fake users, but you can only increase the value of your claimed total liabilities this way. So as long as there aren't customer complaints, this lets you prove a lower bound on your deposits. The trick, of course, is to do all this while respecting the privacy of your users.

To do this we'll use Merkle trees, which we saw in chapter 1. Recall that a merkle tree is a binary tree that's built with hash pointers so that each of the pointers not only says where we can get a piece of information, but also what the cryptographic hash of that information is. The exchange executes the proof by constructing a Merkle tree in which each leaf corresponds to a user, and publishing its root hash. Similar to the naive protocol above, it's each user's responsibility to ensure that they are included in the tree. In addition, there's a way for users to collectively check the claimed total of deposits. Let's delve into detail now.

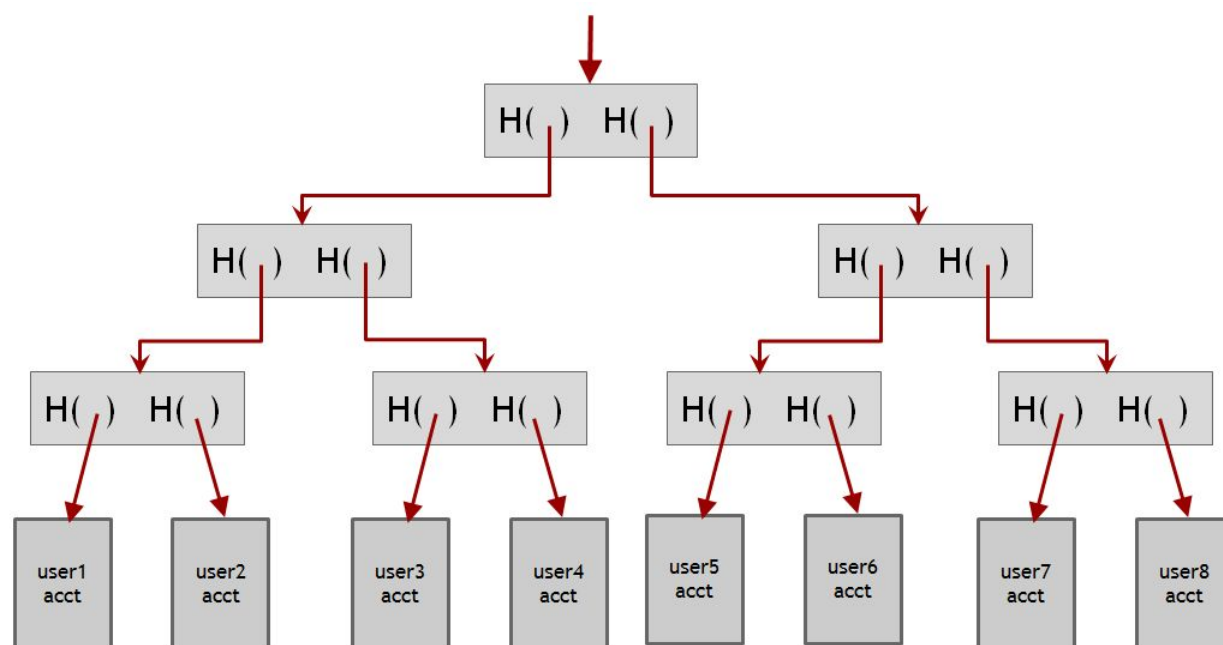


Figure 4.4: Proof of liabilities. The exchange publishes the root of a Merkle tree that contains all users at the leaves, including deposit amounts. Any user can request a proof of inclusion in the tree, and verify that the deposit sums are propagated correctly to the root of the tree.

Now, we're going to add to each one of these hash pointers another field, or attribute. This attribute is a number that represents the total monetary value in bitcoins of all of all deposits that are in the sub-tree underneath that hash pointer in the tree. For this to be true, the value corresponding to each hash pointer should be the sum of the values of the two hash pointers beneath it.

The exchange constructs this tree, cryptographically signs the root pointer along with the root attribute value, and publishes it. The root value is of course the total liabilities, the number we're interested in. The exchange is making the claim that all users are represented in the leaves of the tree, their deposit values are represented correctly, and that the values are propagated correctly up the tree so that the root value is the sum of all users' deposit amounts.

Now each customer can go to the organization and ask for a proof of correct inclusion. The exchange must then show the customer the partial tree from that user's leaf up to the root, as shown in Figure 4.5. The customer then verifies that:

1. The root hash pointer and root value are the same as what the exchange signed and published.
2. The hash pointers are consistent all the way down, that is, each hash value is indeed the cryptographic hash of the node it points to.
3. The leaf contains the correct user account info (say, username/user ID, and deposit amount).
4. Each value is the sum of the values of the two values beneath it.

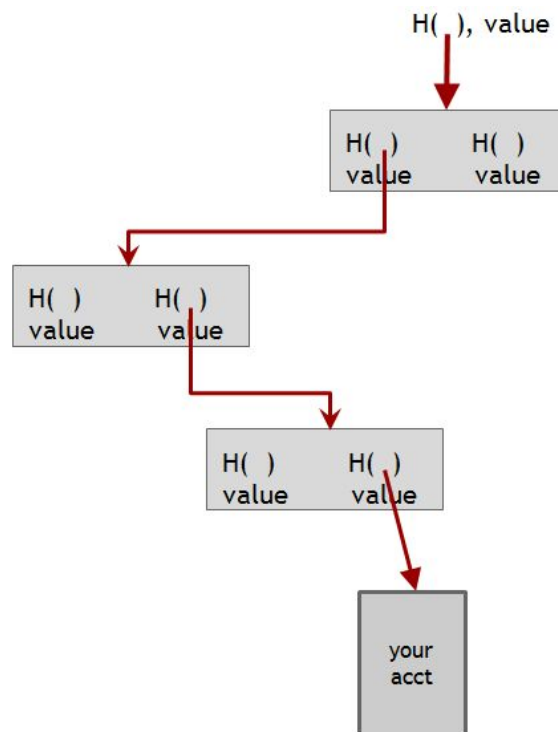


Figure 4.5: Proof of inclusion in a Merkle tree. The leaf node is revealed, as well as the siblings of the nodes on the path from the leaf to the root.

The good news is that if every customer does this, then every branch of this tree will get explored, and someone will verify that for every hash pointer, its associated value equals the sum of the values of its two children. Crucially, the exchange cannot present different values in any part of the tree to different customers. That's because doing so would either imply the ability find a hash collision, or presenting different root values to different customers, which we assume is impossible.

Let's recap. First the exchange proves that they have at least X amount of reserve currency by doing a self transaction of X amount. Then they prove that their customers have at most an amount Y deposited. This shows that their reserve fraction is at least X/Y . What that means is that if a Bitcoin exchange wants to prove that they hold 25% reserves on all deposits — or 100% — they can do that in a way that's independently verifiable by anybody, and no central regulator is required.

This is one aspect of regulation that Bitcoin exchanges can prove voluntarily, but other aspects of regulation are harder to guarantee, as we'll see in Chapter 7.

4.5 Payment Services

So far we've talked about how you can store and manage your bitcoins. Now let's consider how a merchant — whether an online merchant or a local retail merchant — can accept payments in bitcoins in a practical way. Merchants generally support Bitcoin payments because their customers want to be able to pay with bitcoins. The merchant may not want to hold on to bitcoins, but simply receive dollars or whatever is the local fiat currency at the end of the day. They want an easy way to do this without worrying too much about technology, changing their website or building some type of point of sale technology.

The merchant also wants low risk. There are various possible risks: using new technology may cause their website to go down, costing them money. There's the security risk of handling bitcoins — someone might break into their hot wallet or some employee will make off with their bitcoins. Finally there's the exchange rate risk: the value of bitcoins in dollars might fluctuate from time to time. The merchant who might want to sell a pizza for twelve dollars wants to know that they're going to get twelve dollars or something close to it, and that the value of the bitcoins that they receive in exchange for that pizza won't drop drastically before they can exchange those bitcoins for dollars.

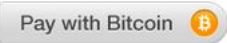

Payment services exist to allow both the customer and the merchant to both get what they want, bridging the gap between these different desires.



Choose A Way To Accept Bitcoin [or see examples](#) of each payment method.

Type ☒ Button ☐ Hosted Page ☐ iFrame ☐ Email invoice

Payment ☒ Buy now ☐ Donation ☐ Subscription

Button Style

☒ 
☐ 

☐ 
☐ 

Item Name Amount

Item Description

Send Funds To

[Show Advanced Options](#)

[Generate Button Code](#)

Figure 4.6: Example payment service interface for generating a pay-with-Bitcoin button. A merchant can use this interface to generate a HTML snippet to embed on their website.

The process of receiving Bitcoin payments through a payment service might look like this to the merchant:

1. The merchant goes to payment service website fills out a form describing the item, price, and presentation of the payment widget, and so on. Figure 4.6 shows an illustrative example of a form from Coinbase.
2. The payment service generates HTML code that the merchant can drop into their website.
3. When the customer clicks the payment button, various things happen in the background and eventually the merchant gets a confirmation saying, “a payment was made by customer ID [customer-id] for item [item-id] in amount [value].”

While this manual process makes sense for a small site selling one or two items, or a site wishing to receive donations, copy-pasting HTML code for thousands of items is of course infeasible. So payment services also provide programmatic interfaces for adding a payment button to dynamically generated web pages.

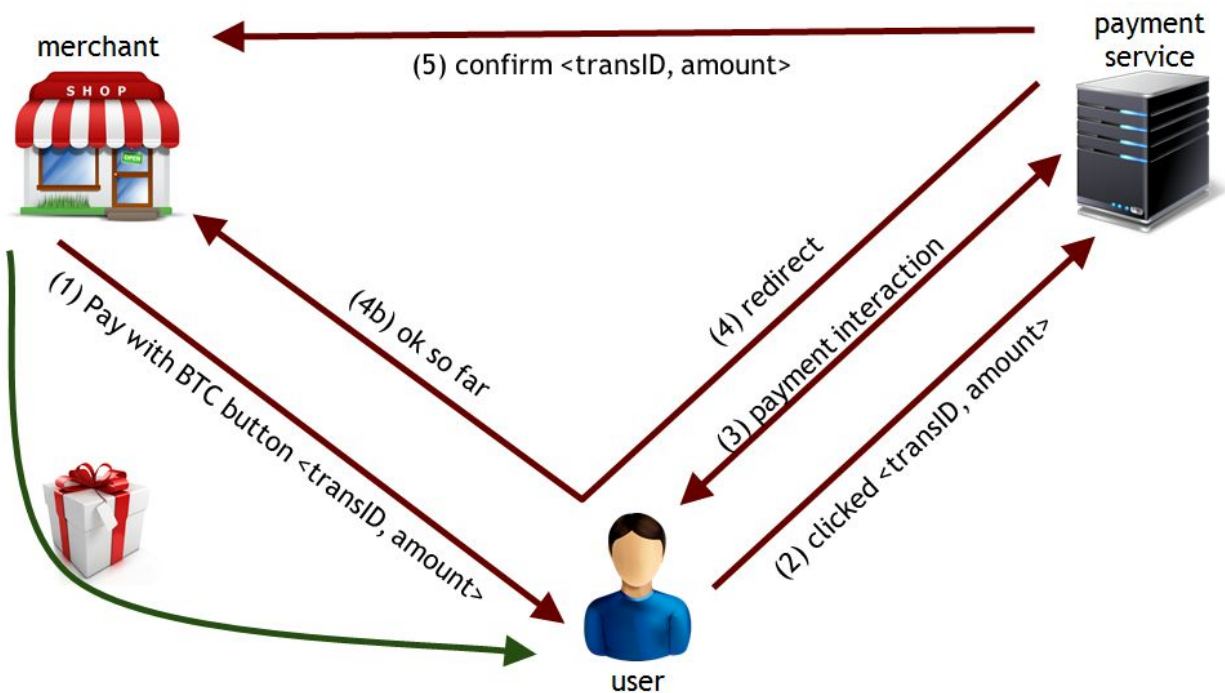


Figure 4.7: Payment process involving a user, merchant, and payment service.

Now let's look at the payment process in more detail to see what happens when the customer makes a purchase with Bitcoin. The steps below are illustrated in Figure 4.7.

1. The user picks out an item to buy on the merchant website, and when it comes times to pay, the merchant will deliver a webpage which will contain the Pay with Bitcoin button, which is the HTML snippet provided by the payment service. The page will also contain a transaction ID — which is an identifier that's meaningful to the merchant and allows them to locate a record in their own accounting system — along with an amount the merchant wants to be paid.
2. If the user wants to pay with bitcoins, they will click that button. That will trigger an HTTPS request to the payment service says that button was clicked, and passing on the identity of the merchant, the merchant's transaction ID, and the amount.
3. Now the payment service knows that this customer — whoever they are — wants to pay a certain amount of bitcoins, and so the payment service will pop up some kind of a box, or initiate some kind of an interaction with the user. This gives the user information about how to pay, and the user will then initiate a bitcoin transfer to the payment service through their preferred wallet.
4. Once the user has created the payment, the payment service will redirect the browser to the merchant, passing on the message from the payment service that it looks okay so far. This might mean, for example, that the payment service has observed the transaction broadcast to the peer-to-peer network, but the transaction hasn't received enough (or any) confirmations so far. This completes the payment as far as the user is concerned, with the merchant's shipment of goods pending a final confirmation from the payment service.

5. The payment service later directly sends a confirmation to the merchant containing the transaction ID and amount. By doing this the payment service tells the merchant that the service owes the merchant money at the end of the day. The merchant then ships the goods to the user.

The final step is the one where the payment service actually sends money to the merchant, in dollars or some fiat currency, via a deposit to the merchant's bank account. This happens at the end of fixed settlement periods, perhaps once a day, rather than once for each purchase. The payment service keeps a small percentage as a fee; that's how they make their revenue. Some of these details might vary depending on the payment service, but this is the general scheme of things.

To recap, at the end of this process the customer pays bitcoins and the merchant gets dollars, minus a small percentage, and everyone is happy. Recall that the merchant wants to sell items for a particular number of dollars or whatever is the local fiat currency. The payment service handles everything else — receiving bitcoins from customers and making deposits at the end of the day.

Crucially, the payment service absorbs all of the risk. It absorbs the security risk, so it needs to have good security procedures to manage its bitcoins. It absorbs the exchange rate risk because it's receiving bitcoins and paying out dollars. If the price of dollars against bitcoins fluctuates wildly, the payment service might lose money. But then if it fluctuates wildly in the other direction the service might earn money, but it's a risk. Absorbing it is part of the payment service's business.

Note that the payment service probably operates at a large scale, so it receives large numbers of bitcoins and pays out large numbers of dollars. It will have a constant need to exchange the bitcoins it's receiving for more dollars so that it can keep the cycle going. Therefore a payment service has to be an active participant in the exchange markets that link together fiat currencies and the Bitcoin economy. So the service needs to worry about not just what the exchange rate is, but also how to exchange currency in large volumes.

That said, if it can solve these problems the fee that the service receives on every transaction makes it a potentially lucrative business because it solves the mismatch between customers' desire to pay bitcoins and merchants' desire to just get dollars and concentrate on selling goods.

4.6 Transaction Fees

The topic of transaction fees has come up in previous chapters and it will come up again in later chapters. Here we'll discuss the practical details of how transaction fees are set in Bitcoin today.

Whenever a transaction is put into the Bitcoin block chain, that transaction might include a transaction fee. Recall from a previous chapter that a transaction fee is just defined to be the difference between the total value of coins that go into a transaction minus the total value of coins that come out. The inputs always have to be at least as big as the outputs because a regular

transaction can't create coins, but if the inputs are bigger than the outputs then the difference is deemed to be a transaction fee, and that fee goes to the miner who makes the block that includes this transaction.

The economics of transaction fees are interesting and we'll come back to this in a later chapter, but for now let's see how transaction fees are actually set in Bitcoin as it operates as of early 2015. These details do change from time to time, but we'll give you a snapshot of the current state.

Why do transaction fees exist at all? The reason is that there is some cost that someone has to incur in order to relay your transaction. The Bitcoin nodes need to relay your transaction and ultimately a miner needs to build your transaction into a block, and it costs them a little bit to do that. For example, if a miner's block is slightly larger because it contains your transaction, it will take slightly longer to propagate to the rest of the network and there's a slightly higher chance that the block will be orphaned if another block was found near-simultaneously by another miner.

So, there is a cost — both to the peer to peer network and to the miners — of incorporating your transaction. The idea of a transaction fee is to compensate miners for those costs they incur to process your transaction. Nodes don't receive monetary compensation in the current system, although running a node is of course far less expensive than being a miner. Generally you're free to set the transaction fee to whatever you want it to be. You can pay no fee, or if you like you can set the fee quite high. As a general matter, if you pay a higher transaction fee it's natural that your transaction will be relayed and recorded more quickly and more reliably.

Current default transaction fees. The current transaction fees that most miners expect is as follows: first of all, no fee is charged if a transaction meets all of these three conditions:

1. the transaction is less than 1000 bytes in size,
2. all outputs are 0.01 BTC or larger
3. priority is large enough

Priority is defined as: $(\text{sum of input age} * \text{input value}) / (\text{transaction size})$. In other words, look at all of the inputs to the transaction, and for each one compute the product of that input's age and its value in bitcoins, and add up all those products. Note that the longer a transaction output sits unspent, the more it ages, and the more it will contribute to priority when it is finally spent.

If you meet these three requirements then your transaction will be relayed and it will be recorded in the block chain without a fee. Otherwise a fee is charged and that fee is about .0001 BTC per 1000 bytes, and as of this writing that's a fraction of a U.S. penny per 1000 bytes. The approximate size of a transaction is 148 bytes for each input plus, 34 bytes for each output plus, and ten bytes for other information. So a transaction with two inputs and two outputs would be about 400 bytes.

The current status quo is that most miners enforce the above fee structure, which means that they will either not service or will service last transactions that don't provide the necessary transaction fees. But there are other miners who don't enforce these rules, and who will record and operate on a transaction even if it pays a smaller fee or no fee at all.

If you make a transaction that doesn't meet the fee requirements it will probably find its way into the block chain anyway, but the way to get your transaction recorded more quickly and more reliably is to pay the standard fee, and that's why most wallet software and most payment services include the standard fee structure in the payments that go on, and so you'll see a little bit of money raked off for transaction fees when you engage in everyday Bitcoin business.

4.7 Currency Exchange Markets

By currency exchange we mean trading bitcoins against fiat currency like dollars and euros. We've talked earlier about services that let you do this, but now we want to look at this as a market — its size, extent, how it operates, and a little bit about the economics of this market.

The first thing to understand is that it operates in many ways like the market between two fiat currencies such as dollars and euros. The price will fluctuate back and forth depending on how badly people want to buy euros versus how badly people want to buy dollars on a particular day. In the Bitcoin world there are sites like bitcoincharts.com that shows the exchange rate with various fiat currencies on a number of different exchanges.

As you'll see if you explore the site, there's a lot of trading going on, and the prices move in real time as trades are made. It's a liquid market and there are plenty of places that you can go to to buy or sell bitcoins. In March 2015 the volume on Bitfinex, the largest Bitcoin — USD exchange, was about 70,000 bitcoins or about 21 million dollars over a 24 hour period.

Another option is to meet people to trade bitcoins in real life. There are sites that help you do this. On localbitcoins.com, for example, you can specify your location and that you wish to buy bitcoins with cash. You'll get a bunch of results of people who at the time of your search are willing to sell bitcoins at that location, and in each case it tells you what price and how many bitcoins they're offering. You can then contact any of them and arrange to meet at a coffee shop or in a park or wherever, give them dollars and receive bitcoins in exchange. For small transactions, it may be sufficient to wait for one or two confirmations on the block chain.

Finally, in some places there are regular meet-ups where people go to trade bitcoins, and so you can go to a certain park or street corner or cafe at a scheduled day and time and there will be a bunch of people wanting to buy or sell bitcoins and you can do business with them. One reason someone might prefer obtaining bitcoins in person over doing so online is that it's anonymous, to the extent that a transaction in a public place can be considered anonymous. On the other hand, opening an account with an exchange generally requires providing government-issued ID due to banking regulation. We'll discuss this in more detail in Chapter 7.

Supply and demand. Like any market, the Bitcoin exchange market matches buyers who want to do one thing with sellers that are willing to do the opposite thing. It's a relatively large market — millions

of U.S. dollars per day pass through it. It's not at the scale of the New York Stock Exchange or the dollar–euro market, which are vastly larger, but it's large enough that there is a notion of a consensus price. A person who wants to come into this market can buy or sell at least a modest amount and will always be able to find a counterparty.

The price of this market, this consensus price, like the price of anything in a liquid market will be set by supply and demand. By that we mean the supply of bitcoins that might potentially be sold and the demand for bitcoins by people who have dollars. The price through this market mechanism will be set to the level that matches supply and demand. Let's dig into this in a little more detail.

What is the supply of bitcoins? This is the number of bitcoins that you might possibly buy in one of these markets, and it is equal to the supply of bitcoins that are in circulation currently. There's a fixed number of bitcoins in circulation. At the time of this writing it's about 13.9 million, and the rules of Bitcoin as they currently stand say that this number will slowly go up and eventually hit a limit of 21 million.

You might also sometimes include demand deposits of bitcoins. That is, if someone has put money into their account in a Bitcoin exchange, and the exchange doesn't keep a full reserve to meet every single deposit, then you'll have demand deposits at that exchange that are larger than the number of coins that the exchange is holding, and depending on what question you're asking about the market it might or might not be correct to include demand deposits in the supply.

So, when should you include demand deposits? Well, basically, you should include demand deposits in a market analysis when demand-deposited money can be sold in that market. For example, if you're talking about exchange of dollars for bitcoins that can happen in an exchange, and the exchange allows demand-deposited bitcoins to be traded for dollars, then they count.

It's worth noting, as well, that when economists conventionally talk about the supply of fiat currency they typically include in the money supply not only the currency that's in circulation — that is, paper and metal money — but also the total amount of demand deposits, and that's for the logical reason that people can actually spend their demand-deposited money to buy stuff. So although it's tempting to say that the supply of bitcoins is fixed at 13.1 million currently or 21 million eventually, for some purposes we have to include demand deposits where those demand deposits function like money, and so the supply might not be fixed the way some Bitcoin advocates might claim. We need to look at the circumstances of the particular market we're talking about in order to understand what the proper money supply is. But let's assume we've agreed on what supply we're using based on what market we're analyzing.

Let's now look at demand. There are really two main sources of demand for bitcoins. There's a demand for bitcoins as way of mediating fiat currency transactions and there's demand for bitcoins as an investment.

First let's look at mediating fiat currency transactions. Imagine that Alice wants to buy something from Bob and wants to pay some money to Bob, and Alice and Bob want to transfer let's say a certain amount of dollars, but they find it convenient to use Bitcoin to do this transfer. Let's assume here that neither Alice nor Bob is interested in holding bitcoins long-term. We'll return to that possibility in a moment. So Alice would buy bitcoins for dollars and transfer them, and once they receive enough confirmations to Bob's satisfaction, he'll sell those bitcoins for dollars. The key thing here from the point of view of demand for bitcoins is that the bitcoins mediating this transaction have to be taken out of circulation during the time that the transaction is going on. This creates a demand for bitcoins.

The second source of demand is that Bitcoin is sometimes demanded as an investment. That is if somebody wants to buy bitcoins and hold them in the hope that the price of bitcoins will go up in the future and that they'll be able to sell them. When people buy and hold, those bitcoins are out of circulation. When the price of Bitcoin is low, you might expect a lot of people to want to buy bitcoins as an investment, but if the price goes up very high then the demand for bitcoins as an investment won't be as high.

A simple model of market behavior. Now, we can do some simple economic modeling to understand how these markets will behave. We won't do a full model here although that's an interesting exercise. Let's look specifically at the transaction-mediation demand and what effect that might have on the price of bitcoins.

We'll start by assuming some parameters. T is the total transaction value mediated via Bitcoin by everyone participating in the market. This value is measured in dollars per second. That's because we assume for simplicity that the people who want to mediate these transactions have in mind a certain dollar value of the transactions, or some other fiat currency that we'll translate into dollars. So there's a certain amount of dollars per second of transactions that need to be mediated. D is the duration of time that bitcoins need to be held out of circulation in order to mediate a transaction. That's the time from when the payer buys the bitcoins to when the receiver is able to sell them back into the market, and we'll measure that in seconds. S is the total supply of bitcoins that are available for this purchase, and so that's going to be all of the hard-currency bitcoins that exist — currently about 14 million or eventually up to 21 million — minus those that are held out by people as long term investments. In other words, we're talking about the bitcoins sloshing around and available for mediating transactions purpose. Finally, P is the price of Bitcoin, measured in dollars per bitcoin.

Now we can do some calculations. First we'll calculate how many bitcoins become available in order to service transactions every second. There are S bitcoins available in total and because they're taken out of circulation for a time of D seconds, every second on average an S/D fraction of those bitcoins will become newly available because they'll emerge from the out-of-circulation state and become available for mediating transactions every second. That's the supply side.

On the demand side — the number of bitcoins per second that are needed to mediate transactions — we have T dollars worth of transactions to mediate and in order to mediate one dollar worth of

transactions we need $1/P$ bitcoins. So T/P is number of bitcoins per second that are needed in order to serve all of the transactions that people want to serve.

Now if you look at a particular second of time, for that second there's a supply of S/D and a demand of T/P . In this market, like most markets, the price will fluctuate in order to bring supply into line with demand. If the supply is higher than the demand then there are bitcoins going unsold, so people selling bitcoins will be willing to lower their asking price in order to sell them. And according to our formula T/P for demand, when the price drops the demand increases, and supply and demand will reach equilibrium.

On the other hand, if supply is smaller than demand it means that there are people who want to get bitcoins in order to mediate a transaction but can't get them because there aren't enough bitcoins around. Those people will then have to bid more in order to get their bitcoins because there will be a lot of competition for a limited supply of bitcoins. This drives the price up, and referring to our formula again, it means that demand will come down until there is equilibrium. In equilibrium, the supply must equal the demand, so we have

$$\frac{S}{D} = \frac{T}{P}$$

which gives us a formula for the price:

$$P = \frac{TD}{S}$$

What does this equation tell us? We can simplify it a bit further: we can assume that D , the duration for which you need to hold a bitcoin to mediate a transaction, doesn't change. The total supply S also doesn't change, or at least changes slowly over time. That means the price is proportional to the demand for mediation as measured in dollars. So if the demand for mediation in dollars doubles then the price of bitcoins should double. We could in fact graph the price against some estimate of the demand for transaction mediation and see whether or not they match up. When economists do this, the two do tend to match up pretty well.

Note is that the total supply S includes only the bitcoins that aren't being held as investments. So if more people are buying bitcoins as an investment, S will go down, and our formula tells us that P will go up. This makes sense — if there's more demand on the investment side then the price that you need to pay to mediate a transaction will go up.

Now this is not a full model of the market. To have a full model we need to take into account the activity of investors. That is, investors will demand bitcoins when they believe the price will be higher in the future, and so we need to think about investors' expectations. These expectations, of course, have something to do with the expected demand in the future. We could build a model that is more complex and takes that into account, but we won't do that here.

The bottom line here is that there is a market between bitcoins and dollars, and between bitcoins and other fiat currencies. That market has enough liquidity that you can buy or sell in modest quantities in

a reliable way, although the price does go up and down. Finally, it's possible to do economic modeling and have some idea about how supply and demand interact in this market and predict what the market might do, as long as you have a way to estimate unknowable things like how much are people going to want to use Bitcoin to mediate transactions in the future. That kind of economic modeling is important to do and very informative, and surely there are people who are doing it in some detail today, but a detailed economic model of this market is beyond the scope of this text.

Further reading

Securing bitcoins has some similarities, as well as important differences, to the way banks secure money. Chapter 10 of Ross Anderson's security textbook, titled "Banking and bookkeeping", is a great read. The entire book is freely available online.

Anderson, Ross. *Security engineering*. John Wiley & Sons, 2008.

The study analyzing closures of Bitcoin exchanges that we referenced:

Moore, Tyler, and Nicolas Christin. *Beware the middleman: Empirical analysis of bitcoin-exchange risk*. Financial Cryptography and Data Security 2013.

Adi Shamir's paper on secret sharing:

Shamir, Adi. *How to share a secret*. Communications of the ACM 22.11 (1979).

Exercises

1. **Proof of reserve.** TransparentExchange claims that it controls at least 500,000 BTC and wants to prove this to its customers. To do this it publishes a list of addresses that have a total balance of 500,000 BTC. It then signs the statement "TransparentExchange controls at least 500,000 BTC" with each of the corresponding private keys, and presents these signatures as proof.

What are some ways in which TransparentExchange might be able to produce such a proof even if it doesn't actually currently control 500,000 BTC? How would you modify the proof to make it harder for the exchange to cheat?

2. **Proof of liabilities.**

TransparentExchange implements a Merkle Tree based protocol to prove an upper bound on its total deposits. (Combined with a proof of reserve, this proves that the exchange is solvent.) Every customer is assigned a leaf node containing an ID which is the hash of her username and a value which is her BTC balance. The protocol specifies that TransparentExchange should propagate IDs and values up the tree by the following recursive definition — for any internal node:

$$\text{node.value} = \text{node.left_child.value} + \text{node.right_child.value}$$

`node.id = Hash(node.left_child.id || node.right_child.id || node.value)`

The exchange publishes the root ID and value, and promises to prove to any customer that her node is included in the tree (by the standard Merkle tree proof of inclusion). The idea is that if the exchange tries to claim a lower total than the actual sum of deposits by leaving some customers out of the tree or by making their node value less than their balance, it will get caught when any of those customers demand a proof of inclusion.

- 2.1. Why can't the exchange include fake customers with negative values to lower the total?
 - 2.2. Show an attack on this scheme that would allow the exchange to claim a total less than the actual sum of deposits.
 - 2.3. Fix this scheme so that is not vulnerable to the attack you identified.
 - 2.4. Ideally, the proof that the exchange provides to a customer shouldn't leak information about other customers. Does this scheme have this property? If not, how can you fix it?
3. **Transaction fees.**
 - 3.1. Alice has a large number of coins each of small value v , which she would like to combine into one coin. She constructs a transaction to do this, but finds that the transaction fee she'd have to spend equals the sum of her coin values. Based on this information (and the default transaction fee policy specified in slide 50), estimate v .
 - 3.2. Can Alice somehow consolidate her coins without incurring any transaction fee under the default policy?
 - 3.3. Compared to a fee structure that doesn't factor the age of the inputs into the transaction fee, what effect might the current default fee structure have on the behavior of users and services?
4. **Multi-signature wallet**
 - 4.1. BitCorp has just noticed that Mallory has compromised one of their servers holding their Bitcoin private keys. Luckily, they are using a 2-of-3 multi-signature wallet, so Mallory has learnt only one of the three sets of keys. The other two sets of keys are on different servers that Mallory cannot access. How do they re-secure their wallet and effectively revoke the information that Mallory has learned?
 - 4.2. If BitCorp uses a 2-out-of-2 instead of a 2-out-3 wallet, what steps can they take in advance so that they can recover even in the event of one of their servers getting broken into (and Mallory not just learning but also potentially deleting the key material on that server)?
5. **Exchange rate**
 - 5.1. Speculate about why buying bitcoins in person generally more expensive than buying from an online exchange.
 - 5.2. Moore and Christin [observe](#) that security breaches and other failures of exchanges have little impact on the Bitcoin exchange rate. Speculate on why this might be.
6. **Payments.** A Bitcoin payment service might receive thousands of payments from various users near-simultaneously. How can it tell whether a particular user Alice who logged into the payment service website and initiated the payment protocol actually made a payment or not?

7. **BitcoinLotto:** Suppose the nation of Bitcoinia has decided to convert its national lottery to use Bitcoin. A trusted scratch-off ticket printing factory exists and will not keep records of any values printed. Bitcoinia proposes a simple design: a weekly run of tickets is printed with an address holding the jackpot on each ticket. This allows everybody to verify the jackpot exists. The winning ticket contains the correct private key under the scratch material.
- 7.1. What might happen if the winner finds the ticket on Monday and immediately claims the jackpot? Can you modify your design to ensure this won't be an issue?
- 7.2. Some tickets inevitably get lost or destroyed. So you'd like to modify the design to roll forward any unclaimed jackpot from Week n to the winner in Week $n+1$. Can you propose a design that works, without letting the lottery administrators embezzle funds? Also make sure that the Week n winner can't simply wait until the beginning of Week $n+1$ to attempt to double their winnings.