

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

*This preface was contributed by Jeremy Clark,
Concordia University.*

Preface — The Long Road to Bitcoin

The path to Bitcoin is littered with the corpses of failed attempts. I've compiled a list of about a hundred cryptographic payment systems, both e-cash and credit card based technologies, that are notable in some way. Some are academic proposals that have been well cited while others are actual systems that were deployed and tested. Of all the names on this list, there's probably only one that you recognize — PayPal. And PayPal survived only because it quickly pivoted away from its original idea of cryptographic payments on hand-held devices!

There's a lot to learn from this history. Where do the ideas in Bitcoin come from? Why do some technologies survive while many others die? What does it take for complex technical innovations to be successfully commercialized? If nothing else, this story will give you an appreciation of how remarkable it is that we finally have a real, working payment mechanism that's native to the Internet.

ACC	CyberCents	iKP	MPTP	Proton
Agora	CyberCoin	IMB-MP	Net900	Redi-Charge
AIMP	CyberGold	InterCoin	NetBill	S/PAY
Allopass	DigiGold	Ipin	NetCard	Sandia Lab E-Cash
b-money	Digital Silk Road	Javien	NetCash	Secure Courier
BankNet	e-Comm	Karma	NetCheque	Semopo
Bitbit	E-Gold	LotteryTickets	NetFare	SET
Bitgold	Ecash	Lucre	No3rd	SET2Go
Bitpass	eCharge	MagicMoney	One Click Charge	SubScrip
C-SET	eCoin	Mandate	PayMe	Trivnet
CAFÉ	Edd	MicroMint	PayNet	TUB
CheckFree	eVend	Micromoney	PayPal	Twitpay
ClickandBuy	First Virtual	MilliCent	PaySafeCard	VeriFone
ClickShare	FSTC Electronic Check	Mini-Pay	PayTrust	VisaCash
CommerceNet	Geldkarte	Minitix	PayWord	Wallie
CommercePOINT	Globe Left	MobileMoney	Peppercoin	Way2Pay
CommerceSTAGE	Hashcash	Mojo	PhoneTicks	WorldPay
Cybank	HINDE	Mollie	Playspan	X-Pay
CyberCash	iBill	Mondex	Polling	

Table 1: Notable electronic payment systems and proposals

Traditional financial arrangements

Back in time before there were governments, before there was currency, one system that worked for acquiring goods was barter. Let's say Alice wants a tool and Bob wants medicine. If each of them happen to have what the other person needs, then they can swap and both satisfy their needs.

On the other hand, let's say Alice has food that she's willing to trade for a tool, while Bob, who has a tool, doesn't have any need for food. He wants medicine instead. Alice and Bob can't trade with each other, but if there's a third person, Carol, who has medicine that she's willing to trade for food, then it becomes possible to arrange a three-way swap where everyone gets what they need.

The drawback, of course, is coordination — arranging a group of people, whose needs and wants align, in the same place at the same time. Two systems emerged to solve coordination: credit and cash. Historians, anthropologists, and economists debate which of the two developed first, but that's immaterial for our purposes.

In a credit-based system, in the example above, Alice and Bob would be able to trade with each other. Bob would give Alice the tool and Bob gets a favor that's owed to him. In other words, Alice has a debt that she needs to settle with Bob some time in the future. Alice's material needs are now satisfied, but she has a debt that she'd like to cancel, so that's her new "want". If Alice encounters Carol in the future, Alice can trade her food for Carol's medicine, then go back to Bob with the medicine and cancel the debt.

On the other hand, in a cash-based system, Alice would buy the tool from Bob. Later, she might sell her food to Carol, and Carol can sell her medicine to Bob, completing the cycle. These trades can happen in any order, provided that the buyer in each transaction has cash on hand. In the end, of course, it's as if no money ever changed hands.

Neither system is clearly superior. A cash-based system needs to be "bootstrapped" with some initial allocation of cash, without which no trades can occur. A credit-based system doesn't need bootstrapping, but the drawback is that anyone who's owed a debt is taking on some risk. There's a chance that the other person never comes back to settle the debt.

Cash also allows us to be precise about how much something is worth. If you're bartering, it's hard to say if a tool is worth more than medicine or medicine is worth more than food. Cash lets us use numbers to talk about value. That's why we use a blended system today — even when we're using credit, we measure debt in the amount of cash it would take to settle it.

These ideas come up in many contexts, especially online systems where users trade virtual goods of some kind. For example, peer-to-peer file-sharing networks must deal with the problem of "freeloaders," that is, users who download files without sharing in turn. While swapping files might

work, there is also the issue of coordination: finding the perfect person who has exactly the file you want and wants exactly the file you have. In projects like MojoNation and academic proposals like Karma, users get some initial allocation of virtual cash that they must spend to receive a file and earn when they send a copy of a file to another user. In both cases, one or more central servers help keep track of users' balances and may offer exchange services between their internal currency and traditional currency. While MojoNation did not survive long enough to implement such an exchange, it became the intellectual ancestor of some protocols used today: BitTorrent and Tahoe-LAFS.

The trouble with credit cards online

Credit and cash are fundamental ideas, to the point that we can sort the multitude of electronic payment methods into two piles. Bitcoin is obviously in the “cash” pile, but let's look at the other one first.

Credit card transactions are the dominant payment method that is used on the web today. If you've ever bought something from an online seller such as Amazon, you know how the arrangement goes. You type in your credit card details, you send it to Amazon, and then Amazon turns around with these credit card details and they talk to the “system”—a financial system involving processors, banks, credit card companies, and other intermediaries.

On the other hand, if you use something like PayPal, what you see is an intermediary architecture. There's a company that sits between you and the seller, so you send your credit card details to this intermediary, which approves the transaction and notifies the seller. The intermediary will settle its balance with the seller at the end of each day.

What you gain from this architecture is that you don't have to give the seller your credit card details, which can be a security risk. You might not even have to give the seller your identity, which would improve your privacy as well. The downside is that you lose the simplicity of interacting directly with the seller. Both you and the seller might have to have an account with the same intermediary.

Today most of us are comfortable with giving out our credit card information when shopping online, or at least we've grudgingly accepted it. We're also used to companies collecting data about our online shopping and browsing activity. But in the 1990s, the web was new, standards for protocol-level encryption were just emerging, and these concerns made consumers deeply uncertain and hesitant. In particular, it was considered crazy to hand over your credit card details to online vendors of unknown repute over an insecure channel. In such an environment, there was a lot of interest in the intermediary architecture.

A company called FirstVirtual was an early payment intermediary, founded in 1994. Incidentally, they were one of the first companies to set up a purely virtual office with employees spread across the country and communicating over the Internet — hence the name.

FirstVirtual's proposed system was a little like PayPal's current system but preceded it by many years. As a user you'd enroll with them and provide your credit card details. When you want to buy something from a seller, the seller contacts FirstVirtual with the details of the requested payment, FirstVirtual confirms these details with you, and if you approve your credit card gets billed. But two details are interesting. First, all of this communication happened over email; web browsers back in the day were just beginning to universally support encryption protocols like HTTPS, and the multi-party nature of payment protocol added other complexities. (Other intermediaries took the approach of encoding information into URLs or using a custom encryption protocol on top of HTTP.) Second, the customer would have ninety days to dispute the charge, and the merchant would receive the money only after three months! Today the merchant does get paid immediately, but, there still is the risk that the customer will file a chargeback or dispute the credit card statement. If that happens, the merchant will have to return the payment to the credit card company.

In the mid '90s there was a competing approach to the intermediary architecture which we'll call the SET architecture. SET also avoids the need for customers to send credit card information to merchants, but it additionally avoids the user having to enroll with the intermediary. In SET, when you are ready to make a purchase, your browser passes your view of the transaction details to a shopping application on your computer which, together with your credit card details, encrypts it in such a way that only the intermediary can decrypt it, and no one else can (including the seller). Having encrypted your data in this way, you can send it to the seller knowing that it's secure. The seller blindly forwards the encrypted data to the intermediary — along with their own view of the transaction details. The intermediary decrypts your data and approves the transaction only if your view matches the seller's view.

SET was a standard developed by VISA and MasterCard, together with many technology heavyweights of the day: Netscape, IBM, Microsoft, Verisign, and RSA. It was an umbrella specification that unified several existing proposals.

One company that implemented SET was called CyberCash. It was an interesting company in many ways. In addition to credit card payment processing, they had a digital cash product called CyberCoin. This was a micropayment system — intended for small payments such as paying a few cents to read an online newspaper article. That meant that you'd probably never have more than \$10 in your CyberCoin account at any time. Yet, amusingly, they were able to get U.S. government (FDIC) insurance for each account for up to \$100,000.

There's more. Back when CyberCash operated, there was a misguided — and now abandoned — U.S. government restriction on the export of cryptography, which was considered a weapon. That meant software that incorporated meaningful encryption couldn't be offered for download to users in other countries. However, CyberCash was able to get a special exemption for their software from the Department of State. The government's argument was that extracting the encryption technology out of CyberCash's software would be harder than writing the crypto from scratch.

Finally, CyberCash has the dubious distinction of being one of the few companies affected by the Y2K bug — it caused their payment processing software to double-bill some customers. They later went bankrupt in 2001. Their intellectual property was acquired by Verisign who then turned around and sold it to PayPal where it lives today.

Why didn't SET work? The fundamental problem has to do with certificates. A certificate is a way to securely associate a cryptographic identity, that is, a public key, with a real-life identity. It's what a website needs to obtain, from companies like Verisign that are called certification authorities, in order to show up as secure in your browser (typically indicated by a lock icon). Putting security before usability, CyberCash and SET decided that not only would processors and merchants in their system have to get certificates, all users would have to get one as well. Getting a certificate is about as pleasant as doing your taxes, so the system was a disaster. Over the decades, mainstream users have said a firm and collective 'no' to any system that requires end-user certificates, and such proposals have now been relegated to academic papers. Bitcoin deftly sidesteps this hairy problem by avoiding real-life identities altogether. In Bitcoin, public keys themselves are the identities by which users are known, as we'll see in Chapter 1.

In the mid 90s, when SET was being standardized, the World Wide Web Consortium was also looking at standardizing financial payments. They wanted to do it by extending the HTTP protocol instead so that users wouldn't need extra software for transactions—they could just use their browser. In fact, they had a very general proposal for how you might extend the protocol, and one of the use cases that they had was doing payments. This never happened -- the whole extension framework was never deployed in any browsers. In 2015, almost two decades later, the W3C has announced that it wants to take another crack at it, and that Bitcoin will be part of that standardization this time around. Given all the past failures, however, I won't be holding my breath.

From Credit to (Crypto) Cash

Now let's turn to cash. We compared cash and credit earlier, and noted that a cash system needs to be "bootstrapped," but the benefit is that it avoids the possibility of a buyer defaulting on her debt. Cash offers two additional advantages. The first is better anonymity. Since your credit card is issued in your name, the bank can track all your spending. But when you pay in cash, the bank doesn't come into the picture, and the other party doesn't need to know who you are. Second, cash can enable offline transactions where there's no need to phone home to a third party in order to get the transaction approved. Maybe later, they go to a third party like a bank to deposit the cash, but that's much less of a hassle.

Bitcoin doesn't quite offer these two properties, but comes close enough to be useful. Bitcoin is not anonymous to the same level as cash is. You don't need to use your real identity to pay in Bitcoin, but it's possible that your transactions can be tied together based on the public ledger of transactions

with clever algorithms, and then further linked to your identity if you're not careful. We'll get into the messy but fascinating details behind Bitcoin anonymity in Chapter 6.

Bitcoin doesn't work in a fully offline way either. The good news is it doesn't require a central server, instead relying on a peer-to-peer network which is resilient in the way that the Internet itself is. In Chapter 3 we'll look at tricks like "green addresses" and micropayments which allow us to do offline payments in certain situations or under certain assumptions.

The earliest ideas of applying cryptography to cash came from David Chaum in 1983. Let's understand this through a physical analogy. Let's say I start giving out pieces of paper that say: "The bearer of this note may redeem it for one dollar by presenting it to me" with my signature attached. If people trust that I'll keep my promise and consider my signature unforgeable, they can pass around these pieces of paper just like banknotes. In fact, banknotes themselves got their start as promissory notes issued by commercial banks. It's only in fairly recent history that governments stepped in to centralize the money supply and legally require banks to redeem notes.

I can do the same thing electronically with digital signatures, but that runs into the annoying "double spending" problem — if you receive a piece of data representing a unit of virtual cash, you can make two (or more) copies of it and pass it on to different people. To stick with our analogy, let's stretch it a little bit and assume that people can make perfect copies and we have no way to tell copies from the original. Can we solve double spending in this world?

Here's a possible solution: I put unique serial numbers into each note I give out. When you receive such a note from someone, you check my signature, but you also call me on the phone to ask if a note with that serial number has already been spent. Hopefully I'll say no, in which case you accept the note. I'll record the serial number as spent in my ledger, and if you try to spend that note, it won't work because the recipient will call me and I'll tell them the note has already been spent. What you'll need to do instead is to periodically bring me all the notes you've received, and I'll issue you the same number of *new* notes with fresh serial numbers.

This works. It's cumbersome in real life, but straightforward digitally provided I've set up a server to do the signing and record-keeping of serial numbers. The only problem is that this isn't really cash any more, because it's not anonymous — when I issue a note to you I can record the serial number along with your identity, and I can do the same when someone else later redeems it. That means I can keep track of all the places where you're spending your money.

This is where Chaum's innovation comes in. He figured out to both keep the system anonymous and prevent double-spending by inventing the digital equivalent of the following procedure: when I issue a new note to you, *you* pick the serial number. You write it down on the piece of paper, but cover it so that I can't see it. Then I'll sign it, still unable to see the serial number. This is called a "blind signature" in cryptography. It'll be in your interest to pick a long, random serial number to ensure that it will most likely be unique. I don't have to worry that you'll pick a serial number that's already been picked — you can only shoot yourself in the foot by doing so and end up with a note that can't be spent.

This was the first serious digital cash proposal. It works, but it still requires a server run by a central authority, such as a bank, and for everyone to trust that entity. Moreover, every transaction needs the participation of this server to go through. If the server goes down temporarily, payments grind to a halt. A few years later, in 1988, Chaum in collaboration with two other cryptographers Fiat and Naor proposed *offline* electronic cash. At first sight this might seem to be impossible: if you try to spend the same digital note or coin at two different shops, how can they possibly stop this unless they're both connected to the same payment network or central entity?

The clever idea is to stop worrying about preventing double-spending and focus on detecting it, after the fact, when the merchant re-connects to the bank server. After all, this is why you're able to use your credit card on an airplane even if there is no network connection up in the skies. The transaction processing happens later when the airline is able to re-connect to the network. If your card is denied, you'll owe the airline (or your bank) money. If you think about it, quite a bit of traditional finance is based on the idea of detecting an error or loss, followed by attempting to recover the money or punish the perpetrator. If you write someone a personal check, they have no guarantee that the money is actually in your account, but they can come after you if the check bounces. Conceivably, if an offline electronic cash system were widely adopted, the legal system would come to recognize double spending as a crime.

Chaum, Fiat, and Naor's idea for detecting double spending was an intricate cryptographic dance. At a high level, what it achieved was this: every digital coin issued to you encodes your identity, but in such a way that no one except you, not even the bank, can decode it. Every time you spend your coin, the recipient will require you to decode a random subset of the encoding, and they'll keep a record of this. This decoding isn't enough to allow them to determine your identity. But if you ever double spend a coin, eventually both recipients will go to the bank to redeem their notes, and when they do this, the bank can put the two pieces of information together to decode your identity completely, with an overwhelmingly high probability.

You might wonder if someone can frame you as a double spender in this system. Say you spend a coin with me, and then I turned around and tried to double-spend it (without redeeming it with the bank and getting a new coin with my identity encoded). This won't work — the new recipient will ask me to decode a random subset, and this will almost certainly not be the same as the subset you decoded for me, so I won't be able to comply with their decoding request.

Over the years, many cryptographers have looked at this construction and improved it in various ways. In the Chaum-Fiat-Naor scheme, if a coin is worth \$100, and you wanted to buy something that cost only \$75, say, there's no way to split that coin into \$75 and a \$25. All you could do is go back to the bank, cash in the \$100 coin, and ask for a \$75 coin and a \$25 coin. But a paper by Okamoto and Ohta uses "Merkle trees" to create a system that does allow you to subdivide your coins. Merkle trees would show up in Bitcoin as well, and we'll meet them in Chapter 1. The Chaum-Fiat-Naor scheme also leaves a lot of room for improvements in efficiency. In particular, the application of something called zero-knowledge proofs to this scheme (most notably by Brands; and Camenisch, Hohenberger,

and Lysyanskaya) was very fruitful—zero-knowledge proofs have also been applied to Bitcoin as we will see in Chapter 6.

But back to Chaum: he took his ideas and commercialized them. He formed a company in 1989 called DigiCash, probably the earliest company that tried to solve the problem of online payments. They had about a five-year head start on other companies like FirstVirtual and CyberCash that we just discussed. The actual cash in DigiCash's system was called Ecash and they had another system called cyberbucks. There were banks that actually implemented it — a few in the US and at least one in Finland. This was in the 1990s, long before Bitcoin, which might come as surprise to some Bitcoin enthusiasts who view banks as tech-phobic, anti-innovative behemoths.

Ecash is based on Chaum's protocols. Clients are anonymous, so banks can't trace how they're spending their money. But merchants in ecash aren't anonymous. They have to return coins as soon as they receive them, so the bank knows how much they're making, at what times, and so on.

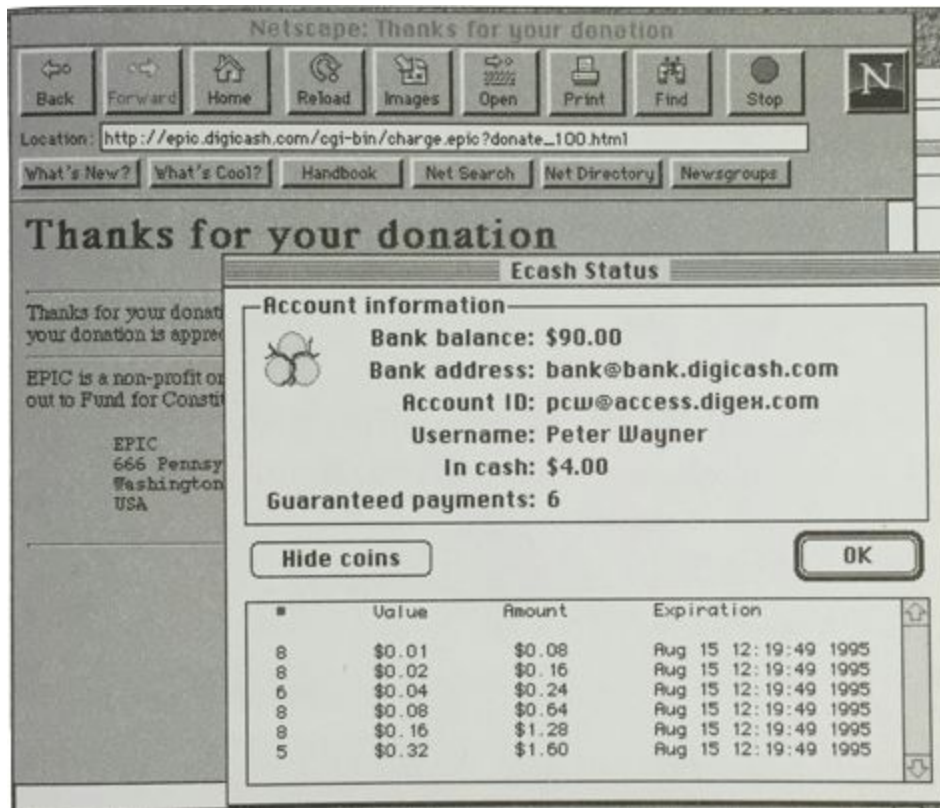


Figure 2: Screenshot of DigiCash

Figure 2 shows a screenshot from the software. As you can see, it shows you your balance as well as all the coins that you have that have been issued to you from the bank. Since there's no way to split your coins, the bank issues you a whole set of coins in denominations of a cent, two cents, four cents,

and so on — powers of two. That way, you (or your software, on your behalf) can always select a set of coins to pay for the exact amount of a transaction.

When you want to make a transaction, say, as in this example, you want to make a donation to the non-profit privacy group EPIC, you'd click on a donation link that takes you to the DigiCash website. That would then open a reverse web connection back to your computer. That means your computer had to have the ability to accept incoming connections and act as a server. You'd have to have your own IP address and your ISP would have to allow incoming connections. If the connection was successful, then the eCash software would launch on your computer and you'd be able to approve the transaction and send the money.

Chaum had several patents on DigiCash technology, in particular, the blind-signature scheme that it used. This was controversial, and it stopped other people from developing eCash systems that used the same protocol. But a bunch of cryptographers who hung out on what was called the cypherpunks mailing list wanted an alternative. Cypherpunks was the predecessor to the mailing list where Satoshi Nakamoto would later announce Bitcoin to the world, and this is no coincidence. We'll talk about the cypherpunk movement and the roots of Bitcoin in Chapter 7.

The cypherpunk cryptographers implemented a version of eCash called MagicMoney. It did violate the patents, but was billed as being only for experimental use. It was a fun piece of software to play with. The interface was all text-based. You could send transactions by email. You would just copy and paste the transactions into your email and send it to another user. Hopefully, you'd use end-to-end email encryption software such as PGP to protect the transaction in transit.

Then there's a proposal called Lucre by Ben Laurie with contributions from many other people. Lucre tries to replace the blind-signature scheme in eCash with a non-patent-encumbered alternative, with the rest of the system largely the same.

Yet another proposal, by Ian Goldberg, tries to fix the problem of not being able to split your coins to make change. His idea was that the merchant could send you coins back if they had some coins, so that you might overpay for the item if you didn't have exact change, and then you'd get some coins back. But notice that this introduces an anonymity problem. As we saw earlier, in eCash, senders are anonymous but merchants aren't. When the merchant sends cash back, technically, they're the sender, so they're anonymous. But you, as someone who has to return this cash to the bank, aren't anonymous. There's no way to design this system without breaking the anonymity of users trying to buy goods. So Goldberg came up with a proposal where there were different types of coins that would allow these transactions to occur, allow you to get change back, and still preserve your anonymity.

Now, why did DigiCash fail? The main problem with DigiCash was that it was hard to persuade the banks and the merchants to adopt it. Since there weren't many merchants that accepted eCash, users didn't want it either. Worse, it didn't support user-to-user transactions, or at least not very well. It was really centered on the user-to-merchant transaction. So if merchants weren't on board, there was

no other way to bootstrap interest in the system. So at the end of the day, DigiCash lost and the credit card companies won.

As a side note, Bitcoin allows user-to-merchant and user-to-user transactions. In fact, the protocol doesn't have a notion of merchant that's separate from the notion of user. The support for user-to-user transactions probably contributed to Bitcoin's success. There was something to do with your bitcoins right from the beginning: send it to other users, while the community tried to drum up support for Bitcoin and get merchants to accept it.

In the later years of the company, DigiCash also experimented with tamper-resistant hardware to try to *prevent* double-spending rather than just detecting it. In this system, you'd get a small hardware device that was usually called a wallet, or some sort of card. The device would keep track of your balance, which would decrease when you spent money and increase if you loaded the card with more money. The point of the device is that there should be no way to physically or digitally go in and tamper with its counter. So if the counter hits zero, then the card stops being able to spend money until it's re-loaded.

There were many other companies that had electronic cash systems based on tamper-resistant hardware. DigiCash later worked with a company called CAFE which was based in Europe. Another company formed around this idea was called Mondex and it was later acquired by Mastercard. Visa also had their own variant called VisaCash.



Figure 3: Mondex system, showing user card and wallet.

Figure 3 shows the user side of the Mondex system. There's a smart card and there's a wallet unit, and you can load either of them with cash. And if you wanted to do user-to-user swap of money, the giver user would first put their card into the wallet and move money off of the card onto the wallet. Then the receiver would stick their card in the wallet then you'd move the money onto the second card. This was a way to exchange digital cash, and it was anonymous.

Mondex trialled their technology in a bunch of communities. One community happened to be a city very close to where I grew up: Guelph, Ontario. You've probably already guessed that it didn't really catch on. A major problem with Mondex cards is that they're like cash — if you lose them or they get stolen, the money's gone. Worse, if there's some sort of malfunction with the card, if the card reader wouldn't read it, there's no way to figure out if that card had balance on it or not. In these scenarios, Mondex would typically eat the cost. They'd assume that the card was loaded and reimburse the user for that lost money. Of course, that can cost a company a lot of money.

Further, the wallet was slow and clunky. It was much faster to pay with a credit card or with cash. And retailers hated having several payment terminals; they wanted just one for credit cards. All these factors together did Mondex in.

However, these cards were smart cards, which means that they have small microcontrollers on them, and that technology has proved successful. In many countries today, including Canada, where I live, every single credit card and every single debit card now has smart card technology in it. It's used for a different purpose, though. It's not used to prevent double-spending — the problem doesn't arise since it's not a cash-based technology. The bank, rather than your card, keeps track of your balance or available credit. Instead the chip is used for authentication, that is, to prove that you know the PIN that's associated with your account. But Mondex was using it long before this technology was adopted widely by the banking industry.

Minting Money out of Thin Air

In the DigiCash system, if you have a digital cash object that's worth \$100, what makes it actually worth \$100? The answer is simple: in order to obtain ecash worth \$100, you'd have to take \$100 out of your bank account and give it to the bank that was issuing you the ecash. But there were a bunch of different proposals for how to do this and different companies did it differently. One far-fetched possibility: what if the government of a particular country actually authorized services to mint digital money, creating new cash out of thin air? That was the idea behind NetCash, although it never got beyond the proposal stage. A different system, used by e-Gold, was to put a pile of gold in a vault and to issue digital cash only up to the value of the gold. Another company called Digigold wasn't fully backed by gold, but had partial reserves.

All of these ideas ultimately peg the value of digital cash to the dollar or a commodity. If the dollar's value goes up or down, the value of your digital money holdings will change along with it. A radically

different possibility is to allow digital money to be its own currency, issued and valued independently of any other currency.

To create a free-floating digital currency that is likely to acquire real value and isn't rocked by volatility in price, you need to have something that's scarce by design. In fact, scarcity is also the reason why gold or diamonds have been used as a backing for money. In the digital realm, one way to achieve scarcity is to design the system so that minting money requires solving a computational problem (or "puzzle") that takes a while to crack. This is what happens in Bitcoin "mining", which we'll look at in Chapter 5.

The basic idea — that solutions to computational puzzles could be digital objects that have some value — is pretty old. It was first proposed by cryptographers Dwork and Naor as a potential solution to email spam back in 1992. What if, every time you sent an email, your computer would have to solve one of these puzzles that would take a few seconds to solve? To enforce this requirement, the recipient's email program would simply ignore your email if you didn't attach the solution to the computational puzzle. For the average user, it wouldn't be that much of a barrier to sending emails because you're not sending emails very frequently. But if you're a spammer, you're trying to send out thousands or millions of emails all at once, and solving those computational puzzles could become prohibitive. A similar idea was later discovered independently by Adam Back in 1997 in a proposal called Hashcash.

These computational puzzles need to have some specific properties to be a useful spam deterrent. First, it should be impossible for a spammer to solve one puzzle and attach the solution to every email he sends. To ensure this, the puzzle should be specific to the email: it should depend on the sender and receiver, the contents of the email, and the approximate time at which it's sent. Second, the receiver should be able to easily check the puzzle solution without having to repeat the process of solving the puzzle. Third, each puzzle should be totally independent of the others, in the sense that solving one puzzle does not decrease the amount of time it takes to solve any other puzzle. Finally, since hardware improves with time and solving any given computational puzzle gets faster and cheaper, recipients should be able to adjust the difficulty of the puzzle solutions that they will accept. These properties can be achieved by using cryptographic hash functions to design the puzzles, and we'll study this in Chapter 1.

Bitcoin uses essentially the same computational puzzle as Hashcash, but with some minor improvements. Bitcoin does a lot more than Hashcash does, though — after all, it takes a whole book to explain Bitcoin! I only mention this because Hashcash inventor Adam Back has said, "Bitcoin is Hashcash extended with inflation control." I think that's overreaching a bit. It's sort of like saying, "a Tesla is just a battery on wheels."

As with any good idea in cryptography, there are many variants of computational puzzles that aim to achieve slightly different properties. One proposal comes from Rivest and Shamir, the R and the S in the RSA cryptosystem. Observe that in Hashcash, your cost to solve a number of puzzles is simply the sum of the individual costs, by design. But this is different from the cost structure for a government to

mint money. If you think about how anti-counterfeiting technology in a paper currency, there's a huge initial cost to acquire all the equipment, create the security features, and so on. But once they've done all that, their costs go down, and it doesn't matter much if they print one bill or a hundred bills. In other words, minting paper money has a huge fixed cost but low marginal cost. Rivest and Shamir wanted to design computational puzzles that would mimic these properties, so that minting the first coin is massively computationally challenging, but minting subsequent coins is a lot cheaper. Their proposal also utilized hash functions, but in a different way. We won't get into the details of their solution, but the problem they were trying to solve is interesting at a high level.

Why did Hashcash never catch on for its intended purpose of preventing spam? Perhaps spam just wasn't a big enough problem to solve. For most people spam as a nuisance, but not something that they want to spend their computing cycles on combatting. We have spam filters today that work pretty well at keeping spam out of our inboxes. It's also possible Hashcash wouldn't have actually stopped spammers. In particular, most spammers today send their spam using 'botnets', large groups of other people's computers that they take control of using malware. They might just as well use those computers to harvest Hashcash. That said, the idea of using computational puzzles to limit access to resources is still an idea that's kicking around. You can see it in some proposals for replacing network protocols, such as MinimalT.

Recording Everything in a Ledger

Another key component of Bitcoin is the block chain: a ledger in which all Bitcoin transactions are securely recorded. The ideas behind the block chain are again quite old, and trace back to a paper by Haber and Stornetta in 1991. Their proposal was a method for secure timestamping of digital documents, rather than an digital money scheme. The goal of timestamping is to give an approximate idea of when a document came into existence. More importantly, timestamping accurately conveys the order of creation of these documents: if one came into existence before the other, the timestamps will reflect that. The security property requires that a document's timestamp can't be changed after the fact.

In Haber and Stornetta's scheme, there's a timestamping service to which clients send documents to timestamp. When the server receives a document, it signs the document together with the current time and as well as a link or a pointer to the previous document, and issues a "certificate" with this information. The pointer in question a special type pointer which links to a piece of data instead of a location. That means that if the data in question changes, the pointer automatically become invalid. In Chapter 1 we'll study how we can create such pointers using hash functions.

What this achieves is that each document's certificate ensures the integrity of the contents of the previous document. In fact, you can apply this argument recursively: each certificate essentially fixes the entire history of documents and certificates up until that point. If we assume that each client in the system keeps track of at least a few certificates — their own documents' certificates, and those of

the previous and following documents — then collectively the participants can ensure that the history cannot be changed after the fact. In particular, the relative ordering of documents is preserved.

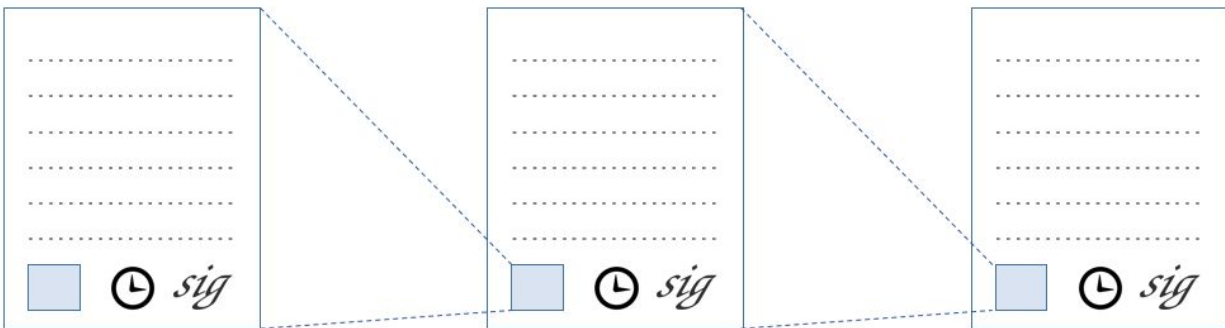


Figure 4: linked timestamping. To create a certificate for a document, the timestamp server includes a hash pointer to the previous document’s certificate, the current time, and signs these three data elements together.

A later paper proposed an efficiency improvement: instead of linking documents individually, we can collect them into blocks and link blocks together in a chain. Within each block, the documents would again be linked together, but in a tree structure instead of linearly. This decreases the amount of checking needed to verify that a particular document appears at a particular point in the history of the system. Visually, this hybrid scheme looks like Figure 5.

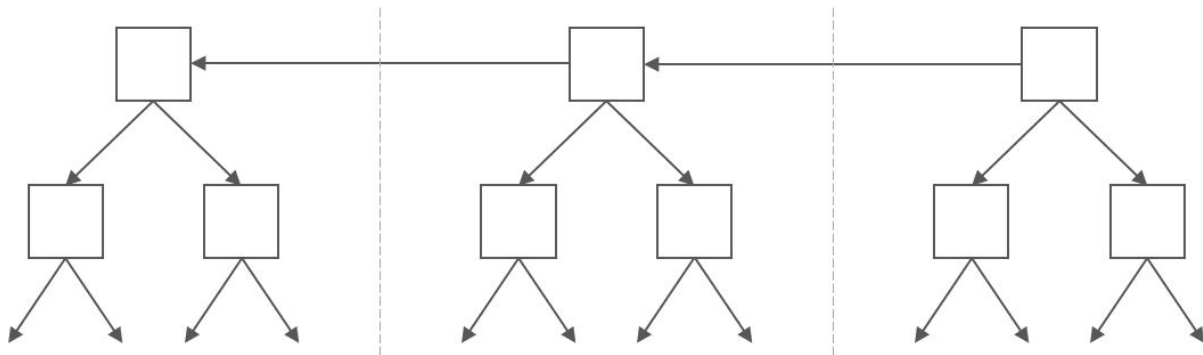


Figure 5: efficient linked timestamping. Arrows represent hash pointers and dotted vertical lines indicate time intervals.

This data structure forms the skeleton of Bitcoin’s block chain, as we’ll see in Chapter 3. Bitcoin refines it a subtle but important way: a Hashcash-esque protocol is used to delay how fast new blocks are added to the chain. This modification has profound and favorable consequences for Bitcoin’s security model. There is no longer the need for trusted servers; instead, events are recorded by a collection of untrusted nodes called “miners”. Every miner keeps track of blocks, rather than having to rely on regular users to do it. Anyone can become a miner by solving computational puzzles to create blocks. Bitcoin also gets rid of signatures, relying only on hash pointers to ensure the integrity of the data structure. Finally, the actual timestamps aren’t of much importance in Bitcoin, and the point of

the system is to record the relative ordering of transactions in a tamper-resistant way. In fact, Bitcoin blocks aren't created in a fixed schedule. The system ensures that a new one is created every 10 minutes on average, but there's considerable variation in the time between successive blocks.

In essence, Bitcoin combines the idea of using computational puzzles to regulate the creation of new currency units with the idea of secure timestamping to record a ledger of transactions and prevent double spending. There were earlier, less sophisticated proposals that combined these two ideas. The first is called b-money, and it was by Wei Dai in 1998. In b-money, anyone can create money using a hashcash-like system. There's a peer-to-peer network, sort of like in Bitcoin. Each node maintains a ledger, but it's not a global ledger like in the Bitcoin block chain. Each node has its own ledger of what it thinks everyone's balance is.

Another similar proposal, by Nick Szabo, is called Bitgold. Szabo says he had the idea for Bitgold as early as 1998, but didn't get around to blogging about it until 2005. The reason I mention this is that there's a minor conspiracy theory popularized by Nathaniel Popper, a New York Times reporter who wrote a very good book on the history of Bitcoin. Popper notes that the blog post timestamps were changed after Satoshi posted the Bitcoin whitepaper so that the Bitgold proposal looks like it was written up about two months after Bitcoin was released. Popper believes, like many other observers, that Szabo could be Satoshi, and he cites the timestamp change as evidence of Szabo/Satoshi trying to cover up the fact that he invented Bitgold before he knew about Bitcoin.

The problem with this explanation is that if you actually read the contents of the blog posts, Szabo is very clear about having had this idea in 1998, and he doesn't try to change those dates. So a more reasonable explanation is that he just bumped the post to the top of his blog after Bitcoin popularized similar ideas, to make sure that people were aware of his prior proposal.

Bitcoin has several important differences from b-money and Bitgold. In those proposals, computational puzzles are used directly to mint currency. Anyone can solve a puzzle and the solution is a unit of money itself. In Bitcoin, puzzle solutions themselves don't constitute money. They are used to secure the block chain, and only indirectly lead to minting money for a limited time. Second, b-money and Bitgold rely on timestamping services that sign off on the creation or transfer of money. Bitcoin, as we've seen, doesn't require trusted timestamping, and merely tries to preserve the relative order of blocks and transactions.

Finally, in b-money and Bitgold, if there is disagreement about the ledger among the servers or nodes, there isn't a clear way to resolve it. Letting the majority decide seems to be implicit in both authors' writings. But since anyone can set up a node — or a hundred, hiding behind different identities — these mechanisms aren't very secure, unless there is a centralized gatekeeper who controls entry into the network. In Bitcoin, by contrast, for an attacker to change history, they must solve computational puzzles at a faster rate than the rest of the participants combined. This is not only more secure, it allows us to quantify the security of the system.

B-money and Bitgold were informal proposals — b-money was a post on a mailing list and Bitgold was a series of blog posts. Neither took off, or was even implemented directly. Unlike the Bitcoin white paper, there wasn't a full specification or any code. The proposals gloss over issues that may or may not be solvable. The first, as we've already mentioned, is how to resolve disagreements about the ledger. Another problem is determining how hard the computational puzzle should be in order to mint a unit of currency. Since hardware tends to get dramatically cheaper over time for a fixed amount of computing power, Bitcoin incorporates a mechanism to automatically adjust the difficulty of the puzzles periodically. B-money and Bitgold don't include such a mechanism, which can result in problems since coins may lose their value if it become trivially easy to create new ones.

Hints about Satoshi

You may know that Satoshi Nakamoto is the pseudonym adopted by the creator of Bitcoin. While his identity remains a mystery, he communicated extensively in Bitcoin's early days. Let's use this to dig a little bit into questions like when he started working on Bitcoin, to what extent he was influenced by the prior ideas we've looked at, and what motivated him.

Satoshi says he started coding Bitcoin around May 2007. I'll take him at his word; the fact that he's anonymous is not a reason to think he'd lie about things like that. He registered the domain bitcoin.org in August 2008. And at that time, he started sending private emails to a few people who he thought might be interested in the proposal. Then a little later in October 2008, he publicly released a white paper that described the protocol, and then soon after, he released the initial code for Bitcoin as well. Then he stuck around for about two years, during which he posted lots of messages on forums, emailed with lots of people, and responded to people's concerns. On the programming side, he submitted patches to the code. He maintained the source code in conjunction with other developers, fixing issues as they arose. By December 2010, others had slowly taken over the maintenance of the project, and he stopped communicating with them.

I've been referring to Satoshi Nakamoto as a "he," but I have no particular reason to believe Satoshi is a man and not a woman. I'm just using the male pronoun since Satoshi is a male name.

I've also been referring to him as a single individual. There is a theory that Satoshi Nakamoto might be a collection of individuals. I don't buy this theory — I think Satoshi is probably just one person. The reason is that if we look at the entirety of the online interactions undertaken under the Satoshi pseudonym, if we think about the two years that Satoshi spent replying to emails and patching code, it's hard to imagine that this could be multiple people sharing user accounts and passwords, responding in a similar style and a similar voice, and making sure they didn't contradict each other. It just seems a much simpler explanation that at least this portion of Satoshi's activity was done by a single individual.

Furthermore, it's clear from his writings and patches that this individual understood the full code base of Bitcoin and all its design aspects. So it's very reasonable to assume that the same individual wrote

the original code base and the white paper as well. Finally, it's possible that Satoshi had help with the original design. However, after Bitcoin's release, we can see for ourselves that Satoshi was quick to attribute any help he received from other contributors. It would be out of character for him to mislead us about inventing something by himself if he had had help from other people.

Next, we might ask ourselves, "What did Satoshi know about the history of ecash?" To understand this better, we can start by looking at what he cites in his white paper as well as the references that existed on early versions of the Bitcoin website. In the white paper he cites some papers on basic cryptography and probability theory. He also cites the time-stamping work that we saw earlier, and it's very natural to think that he based the design of the block chain on these references since the similarities are so apparent. He also cites the Hashcash proposal whose computational puzzle is very similar to the one that's used in Bitcoin. He also has a reference to b-money. Later, on the website, he added references to Bitgold and as well to a scheme by Hal Finney for reusing computational puzzle solutions.

But, if we look at the email exchanges that were made public by people who corresponded with Satoshi Nakamoto in the early days, we find that the b-money proposal was actually added after-the-fact, at the suggestion of Adam Back. Satoshi then emailed Wei Dai who created b-money and apparently, Dai was the one that told him about Bitgold. So these proposals weren't probably inspirations for the original design. He later corresponded a lot with Hal Finney, and that's quite a reasonable explanation for why he cites Finney's work, at least on the website.

Based on this, it seems plausible that when creating Bitcoin, Hashcash and time-stamping were the only things from the history of ecash that Satoshi knew about or thought were relevant. After he came to know of b-money and Bitgold, however, he seems to have appreciated their relevance. In mid-2010, the Wikipedia article on Bitcoin was flagged for deletion Wikipedia's editors because they thought it wasn't noteworthy. So there was some discussion between Satoshi and others about how to word the article so that Wikipedia would accept it. To that end, Satoshi suggested this description of Bitcoin: "Bitcoin is an implementation of Wei Dai's b-money proposal on Cypherpunks in 1998 and Nick Szabo's Bitgold proposal." So Satoshi, by this point, did see positioning Bitcoin as an extension of these two ideas or an implementation of these two prior systems as a good explanation of how it worked.

But, what about everything else — the Chaumian ecash schemes and the credit card proposals that we looked at? Did Satoshi know any of that history when designing Bitcoin? It's hard to tell. He didn't give any indication of knowing that history, but it's just as likely that he didn't reference this because it wasn't relevant to Bitcoin. Bitcoin uses a completely different decentralized model and so there's no compelling reason to dwell on old centralized systems that failed.

Satoshi himself makes this point, by mentioning Chaumian ecash in passing, in one of his posts to the Bitcoin forums. Writing about another proposal called opencoin.org, he notes that they seem to be "talking about the old Chaumian central mint stuff, but maybe only because that was the only thing available. Maybe they would be interested in a new direction. A lot of people automatically dismiss

e-currency as a lost cause because of all the companies that failed since the 1990's. I hope it's obvious it was only the centrally controlled nature of those systems that doomed them. I think this is the first time we're trying a decentralized, non-trust-based system." That gives us a pretty good idea what Satoshi thought of the earlier proposals, and specifically how he felt Bitcoin was different. Bitcoin's decentralization is indeed a defining feature that sets it apart from almost everything we've look at.

Another interesting quote from Satoshi suggests that he might not be an academic. Most academic researchers think about ideas and write them down immediately, before they build the system. Satoshi says that he took an opposite approach: "I actually did Bitcoin kind of backwards. I had to write all the code before I could convince myself that I could solve every problem, then I wrote the paper. I think I will be able to release the code sooner than I could write a detailed specification."

Since there's bit of myth around Satoshi, it's worth mentioning that he made mistakes like everyone else and that wasn't a perfect oracle of the future. There are bugs and questionable design choices in the original Bitcoin code as well as in its design. For example, there was a feature to send bitcoins to IP addresses that never caught on and, in retrospect, was a bad idea. When he described what Bitcoin was useful for, his scenarios were centered on the idea of using it across the internet. That use case is central to Bitcoin, of course, but it's not the only one. He didn't indicate a vision of going into a coffee shop and being able to pay for your coffee with Bitcoin, for example.

A final question we may ask ourselves, colored by what we understand from the history of digital cash, is, "Why does Satoshi maintain his anonymity?" There are many possible reasons. To begin with, it might be just for fun. Many people write novels anonymously, and there are graffiti artists like Banksy who maintain their anonymity. In fact, in the community that he was involved in at that time, the Cypherpunk community and the cryptography mailing list, it was common practice for people to post anonymously.

On the other hand, there could have been legal worries behind Satoshi's choice. Two U.S. companies, Liberty Reserve and e-Gold, ran into legal trouble for money laundering. In 2006, one of the founders of Liberty Reserve fled the United States, fearing that he would be indicted on money laundering charges. E-Gold's founders, on the other hand, stayed in the United States, and one was actually indicted and eventually pled guilty to the charges. This guilty plea was registered just right before Satoshi set up the Bitcoin website and started emailing people about his proposal. That said, numerous people have invented ecash systems, and nobody else was scared of the legal implications or has chosen to remain anonymous. So this may have been the reason, it may not have been the reason.

It's also worth recalling that certain aspects of ecash were patented, and that members of the Cypherpunk movement were concerned about implementing ecash systems due to these patents. In fact, one post to the cypherpunks mailing list proposed that a group of anonymous coders implement ecash so that if someone were to sue, they wouldn't be able to find the coders. While it is difficult to think that Bitcoin would violate the ecash patents given how different its design is, perhaps Satoshi

was being extra cautious. Or maybe he was just inspired by the idea of an anonymous coder from the cypherpunk community.

A final reason that's often cited is personal security. We know that Satoshi has a lot of bitcoins from his mining early on, and due to Bitcoin's success these are now worth a lot of money. I think this is a plausible reason. After all, choosing to be anonymous isn't a decision you make once, it's something that you do on a continual basis. That said, it probably wasn't Satoshi's original reason. The first time Satoshi used the name Satoshi Nakamoto, he hadn't even released the whitepaper or the codebase for Bitcoin, and it's hard to imagine that he had any idea that it would be as successful as it was. In fact, at many points in its early history, Satoshi was optimistic but cautious about Bitcoin's prospects. He seems to have understood that many previous efforts had failed and that Bitcoin might fail as well.

Concluding remarks

The success of Bitcoin is quite remarkable if you consider all the ventures that failed trying to do what it does. Bitcoin has several notable innovations including the block chain and a decentralized model that supports user-to-user transactions. It provides a practically useful but less-than-perfect level of anonymity for users. In Chapter 6 we'll take a detailed look at anonymity in Bitcoin. In one sense it's weaker than the strong anonymity in DigiCash, but in another sense it's stronger. That's because in DigiCash, it was only the senders of the money that maintained their anonymity, and not the merchants. Bitcoin gives both senders and merchants (whether users or merchants) the same level of anonymity.

Let me conclude with some lessons that we can learn from Bitcoin through the lens of the previous systems that we've looked at. The first is to not give up on a problem. Just because people failed for 20 years in developing digital cash doesn't mean there isn't a system out there that will work. The second is to be willing to compromise. If you want perfect anonymity or perfect decentralization you'll probably need to worsen other areas of your design. Bitcoin, in retrospect, seems to have made the right compromises. It scales back anonymity a little bit and requires participants to be online and connected to the peer-to-peer network, but this turned out to be acceptable to users.

A final lesson is success through numbers. Bitcoin was able to build up a community of passionate users as well as developers willing to contribute to the open-source technology. This is a markedly different approach than previous attempts at digital cash, which were typically developed by a company, with the only advocates for the technology being the employees of the company itself. Bitcoin's current success is due in large part to the vibrant supporting community who pushed the technology, got people using it, and got merchants to adopt it.