Duality: An excerpt

It was inevitable.

"The Times 03/Jan/2009 Chancellor on brink of second bailout for banks."

I think the question that is most often brought up about bitcoin, is why. Why was it created at all? And the answer, as plainly visible as anyone could see, I embedded in the coinbase of the *first* genesis block.

In creating bitcoin, much thought went into how to integrate already existing ideas from the cypherpunk community. Ideas proposed by individuals like Adam Back, Wei Dai, David Chaum, and Hal Finney.

David Chaum preceded me by almost twenty years, but with his paper Untraceable Electronic Cash he explored the possibility of anonymous transactions using a number of cryptographic protocols. Its inherent flaw however, was that it was centralized. And like all things, people can lose trust in something that is controlled by one authority.

Still, it paved the way in showing us (as in cypherpunks) at least that anonymous transactions *were* possible.

The principles for bitcoin originated from the cypherpunks, a community I naturally gravitated to as a fourteen year old, a place where anonymity was as fundamental as breathing, where in order for genuine freedom of speech to exist in an open society one had to be able to fully and anonymously express themselves.

(I'm going to take a moment here to explain something. I know that some of you might be reading this or hearing about it for the first time, might not know, so I should state it publicly, although by now it is assumed, but has never been publicly stated before, so I will make it official.)

Satoshi Nakamoto is **not** a real name. Specifically, not a legal name.

It is primarily the essence of thoughts and reason.

I wanted the most common name, which I knew no one outside of Japan had any recollection that Satoshi Nakamoto, was the equivalent of "John Smith."

It took time for the public to come to this conclusion, but most with direct access to me had figured it out long ago.

Moving on.

In April of 2009 is when Mike Hearn first emailed asking about the project. For all intents and purposes, Mike appeared to me as someone who knew what he was talking about but nonetheless was eager to learn new things. His curiosity piqued me as someone inquisitive, asking me whether bitcoin was based

on one “global chain” or many, which now most would refer to as the “*blockchain*.” I pointed out how it was all part of one global chain, with all blocks forming part of that chain.

I don't think anyone knows this, but the word blockchain did **not** come into play until after the fact. *Prebitcoin*, it was referred to as, the **timechain**. That is because it wasn't about the blocks in the beginning, but rather about time, specifically the precise intervals of time upon which the blocks were released. Before they became hardcoded to the rest of the chain. In the finalized version, version 2, this was changed to *blockchain* as it became more obvious to me that blocks were the underlying element linked in the chain, not only through time. That is how the public would perceive it at least. You see, Bitcoin, the one you see, the one you have bought or sold coins from, is actually, version **2**. Similarly, the word *fork* didn't come into play until after the fact either, *prebitcoin* this was called the **branch point**.

Mike also brought up issues of scaling, and in reference to his concerns, I used the example of the Visa network, which at the time handled 15 million transactions a day (now handles roughly ten times that). Bitcoin was already able to scale much larger than this. Or so I thought.

Bitcoin is able to supplant centralized networks in various ways. This is why some incumbents feared bitcoin, is not because of the wow factor, but because of its network advantages. In terms of speed and security, bitcoin was superior. Here was, for the first time, a distributed network of nodes that validated one immutable record (the blockchain), which in this case happened to be a form of currency. Even if one of these nodes came down, the chain would still continue as it was.

I want to state something that I am sure Mike will agree with me on, and that is that Bitcoin started as most things do, a love, an idea, a dream. You see, bitcoin is just one of the few that made it, and it is quite clear to me now why. Bitcoin started off as a very small project on sourceforge with small but potential for grand ambitions. What I mean by this is that bitcoin is just one of the tens of thousands of projects. It is not special in this regard.

Why did it succeed? I attribute it to a few factors. I view the fact that something like the blockchain, which had already been done in a similar fashion but without much fanfare, was not the prime reason, rather it was because of the very nature and premise of bitcoin. Here, for once, was this idea that you could generate your own form of money. That's the primary and sole reason, is because it was related to this thing called *money*. It wasn't about the profficiency of the code or the novelty, it was because it had to do with money. It centered around *money*.

That is something people cared about.

After all, plenty of projects on Sourceforge at the time were just as well coded, well maintained, if not better, by teams, and even if someone else had created the blockchain before me, had it been used for something else beyond currency, it probably would not have had much of an outcome.

I will put this into effect, look at the projects using the blockchain today and which are the most successful? Which are most popular, most utilized? All of those that center around *money*.

So it is by no surprise then that bitcoin took off the way it did. At least not now. But for the last decade, yes it was hard to reason why it succeeded when other projects had not. I refer to the thousands upon thousands of projects that preceded it and have since surpassed it.

Back on topic.

In referring back to Moore's Law, I really did believe that computers would be equipped with hardware that by the time I write this, would be a 100 times faster than it was ten years ago. I was wrong.

In truth, hardware did not advance as quickly as I had anticipated and actually did me a disservice. I learned then to not speculate about the future. Because you usually get it wrong.

Mike was one of the few individuals who saw the likelihood of ASICS (application specific integrated circuits) and he had smartly anticipated the possibility, as did I, of specialized hardware eventually being used to mine for bitcoins. ASICS by definition are custom chips built for one singular purpose, in this case, to mine for bitcoins. Early on at least, I wanted it to be possible for anyone using a standard PC to mine for bitcoins. But as I soon realized, people already had begun to find ways to game the system by using specialized hardware. Still though, for the better part of a year, I was mostly the only one using the network on a consistent basis, mining the coins myself.

(I am going to interject here. From the very beginning, I was what could be considered the first "server farm" in the broadest sense of the word by the sheer number of computers, which at that point anything more than two was enough.)

I suspected that eventually most nodes would be using their graphical units by the time it became feasible to, to mine for bitcoin versus using general purpose CPUs. And it made complete sense if your goal was to get as many as possible. Why use a CPU when you could have a GPU do the work that much quicker? That (CPUs) is what most users had at their disposal at the time gave me reason to neglect mentioning that there were more sophisticated options out there, so I started with that. I was aware of it (ASICS) yes, still I didn't really want to start an arms race for a network with few users, but I also didn't care how someone decided to mine for them, whatever method they chose, I left it up to the individual to figure it out. Although I *did* want users to use their PCs and for it to stay that way for a while. Still, some had started to see that bitcoin was accruing *value* and the focus started to shift…

I think Mike's real concern was how secure bitcoin could be and whether it was a truly decentralized protocol, and what incentive users had to commit to that protocol. I mentioned to him that the fee rate was dictated by the nodes themselves, but bitcoin itself could work without fees at all.

(I want to take a moment here to clarify something)

This is still an area of heated discussion**, transaction fees**. Prebitcoin, I had decided to include the transaction fee, and I had the fee set at **1 CENT** (to give you perspective, one bitcoin today is a hundred million cents or 'satoshi', so, a ridiculously small amount) but I knew I wanted to change this later on and make it a user option setting, with the optional fee being zero to start. In hindsight, maybe I should have kept this as transaction fees seem to have skyrocketed and become skewed, almost to the point that it has become redundant, but in retrospect, for that very same reason people now are working on making transactions feeless at scale (scale means *x-illions*, replace x with any prefix you want, bi, tri, etc.).

At some point, it is probable bitcoin will not need to depend on transaction fees at all. Fees could someday cease to be used as an incentive. Much like the US dollar is used by over 60% of the worlds population, it all boils down to one thing--trust. If you replaced the US dollar with the Yen or the Pound tomorrow, it would only mean that people trust it more, and will use it for that reason. The same could happen with bitcoin. If it becomes popular enough, it too has the same potential to be considered a widely used, adopted, and trusted form of currency.

Mike never really delved into who I was, more concerned with the prospects of the technology than the person (or persons) behind it. For that reason, I very much respected his concerns and made sure to answer them to the best of my ability.

Subsequent discussions centered on the use of micropayments (not yet practical, I will go into detail in the book), and how it made no sense for an attacker to try to fool the network, for honest nodes would broadcast the transaction and it would soon be recorded into the blockchain by other nodes, disallowing an attacker from any attempt at double spending. This to me, is the fundamental problem bitcoin was able to solve, which up to that point, hadn't been figured out—was the issue of double spending.

He did though show enthusiasm about what bitcoin could be used for, and although some of my half-baked ideas never made it through (i.e. a marketplace, an escrow system) I did make a mention of it. In the first version (prebitcoin), I envisioned it could be used for a marketplace, with user reviews and comments. Think of a decentralized eBay that no one controlled. Bitcoin did have more potential than just as a form of currency. Something he suggested implementing was comments for indirect transfers. I pointed out that this was not possible.

Unfortunately, the way bitcoin was designed did not allow for this. The EC-DSA that bitcoin uses only allowed for one thing, to verify signatures, which unlike RSA can't be used to encrypt messages. In planning for the future, I had to make the choice between doing what was practical, siding with EC-DSA which was fundamental in making the block chain compact enough to be practical for the future. I didn't choose what would be good for the time, but rather I made the choice to implement what would be good for the long term.

In the end, it turned out the verification times, not the size was the drawback. With RSA keys too huge, I took a leap of faith on EC-DSA and upped the key size in order to be 256 bit compliant.

But for all the right things people say I did, bitcoin did not come without its flaws. Flaws of course which could be fixed but this taught me a valuable lesson: nothing is perfect, and bitcoin proved to be of no exception.

People sometimes view bitcoin as an overnight success, but really it was the better part of two years, devoting many long hour chunks to it, sectioning off blocks a day to the project, while (barely) maintaining a productive and healthy way of life. It was challenging. But it was fulfilling. This goes to show, nothing is an overnight success.

People may forget about this fact now but for the first year, it really was mostly myself who was the sole and active participant in the network. I was both maintaining it, using it, making changes to the code, fixing bugs, and promoting its use. Most people in the beginning were in fact just installing bitcoin once and never using it again, circumventing its intended use. The only person that really chugged away and stayed with it that first year, was Hal Finney.

One of Mike's concerns centered on non-reversible transactions. Bitcoin was fundamentally designed for non-reversible transactions. Nevertheless, bitcoin could do much more than this, with the network infrastructure capable of a variety of features, not the least of which was an escrow system.

Other concerns which repeatedly come up even today is the size of the blocks, which I can say, was an unintended outcome that no one could have easily anticipated in the beginning but of which a solution

could be implemented quite easily. What it will take is consensus, which is always hard to reach (more so now it seems), but the size of blocks in the end would not matter much on the client side. Once mining pools are the only ones resorting to mining, most on the client side will not be affected and are in effect just buying or selling bitcoin. Most users today are not actively running as a node either way.

And although Mike never really was interested in who I was, there were others that did wonder and ask questions, some reasons that came up--my english was without error, another that I never wrote anything in japanese, the list goes on. Few had suspected (in the beginning at least) that it could not have possibly been the work of just one person, that, perhaps it was a pseudonym, and in that regard...they were right.

Gavin was one of those people.

In June 2010 is when Gavin introduced himself, coming off as an enthusiastic and skillful programmer who liked to take initiative and wanted bitcoin to succeed from day one. From the time of release and even to the moment I left the community and had already decided to move on, I never made it a point to talk about myself. I responded to technical questions, all other (personal) questions I sidestepped. Cypherpunks and those from the cryptography community knew not to ask these kinds of questions, but to some programmers (not pointing fingers here), some of which used their real names, this was a question they felt could be brought up.

I made it known that I was available to answer technical questions related to bitcoin. For the obvious reason that bitcoin wasn't about *me*, and really I had come up with the name for the superfluous fact that I needed one. You see, I couldn't quite publish an academic style paper without some skepticism by the incumbents, contrary to what the Cyperphunks preached, most of the contributions made by people in the cryptography community were made by people using their real names. By the time Gavin proceeded to ask me how old I was and whether Satoshi was my real name…

(*The continuation of this story along with the origins of the name in the book*)

By this point, it was starting to become obvious to some in the community that Satoshi Nakamoto might not be real. And although I made every effort to simply ignore these claims, some people were clever enough to see through it. Perhaps Satoshi Nakamoto was a pseudonym after all. Perhaps too I wasn't from Japan, but rather someone of British descent. Some came to these conclusions after analyzing my writing style , which to be fair, is something that was taught to me very early on--proper English.

(I must admit though, that I notice my style of writing has changed slightly in the years since..I don't double space like before, I also notice that I take myself much less serious now, probably attributed to age. Trying as a twenty something to appear older than what you are in a community that counts age as experience in order to throw people off and get them to accept your work, while trying to push something you so strongly believe in, oh the things you'll do in order to be taken more serious...)

Back to where I was.

In typical Japanese fashion, names are written by Last name first followed by First name. This contradicted the way I had employed the name, writing First name first and Last name last as taught to me in my youth. This corresponded with someone who had been taught Standard Proper English and was by definition, a native speaker.

What didn't come up (at the time), which I found odd, was the question of whether Satoshi Nakamoto was just one, or a group of us.. Here is where I will stop. But the truth is one that people will not come to expect. Because the truth is too special to give away, requires a long answer, which will be in the book.

I will say this though, consider for a moment the distinction; as to whether I had help or was part of that help in creation, and then separate that from the person who followed, which for the most part, was very consistent.

Other clues could not be ignored. For the first year, I had ran the network from my own computers (I also was a university researcher working out of a lab at the time--I might explain further in the future) which meant I needed anonymizing software to mask my location, but even with all these protective layers, I had neglected one simple fact.

When I set up for the public announcement of bitcoin, I took these preemptive measures to mask my location, but even with all of these measures in place, I did not account for one thing that could give my location away, *timestamps*. Some were intuitive enough to graph together the hours in which I would post on the forums and commit to the repository and formed a literal 'map' of when I was awake and when I was asleep (disturbing to say the least), a graph which placed my sleeping habits between the hours of 1 am to 7 am. The timing of sleeping patterns then correlated on weekends. It was then clear to see, that I must be located somewhere on the east coast (unless of course, I were a night owl). This meant that Satoshi could be anyone on the eastern time zone which comprised the most populated coastal area of the US, or one out of 112,642,503 individuals, roughly.

These two factors, my writing style and the times I would actively be contributing on the forums, responding to emails or committing to the repository, were things out of my control. Nonetheless, they placed me in a certain predicament, making it obvious that I must be located somewhere on the east coast even when I had made sure to always use British date and time notations, which meant thousands of miles separated me from any Japanese or British connection. Indeed, the article used from the Times was meant to throw off would be cypherpunks, but even so it was quite clear now, that Satoshi at least, was residing somewhere on the Eastern Time Zone.

My mother who is an author (albeit small circulation--and who indirectly inspired me to share this glimpse of time) is the one responsible for my style of writing. I got my punctuation from her, as she not only took it upon herself to teach me how to read, but also how to write. I also came from a family where my Grandmother had started a small publishing "company" (really consisting solely of her and my mother), and my idea of fun at an early age was finding mistakes and errors in words. It is by no coincidence then that I gravitated towards spelling, going so far as to become the spelling bee runner up in the fourth grade. It may seem silly to some, but this is something I still take great honor in. And although I never followed through on this, I still take joy in finding mistakes, be it in code or in writing.

There are questions that always seem to come up, either about me or about the way bitcoin was designed. Questions around the choice of encryption, the block size, the supply, the programming language of choice. I'll start with the most obvious question and then try to answer the others throughout these pages.

For starters, many may wonder what the reasoning behind the fixed supply is. Why **21** million? The truth is, it was **an educated guess**. The math worked out, or as close to it as I had wanted it to. Before settling

on 21 million however, I had considered making **100 BTC** as the reward, and **42**—the answer to life, the universe, and everything. But afraid that others would consider my reference to Hitchhikers Guide to the Galaxy a quip and at the expense of not being taken seriously, I changed it to 21 million.

It worked out to an even 10 minutes per block:

21000000 / (50 BTC * 24hrs * 365days * 4years * 2) = 5.99 blocks/hour

I rounded it to 364.583 days/year to be more precise. The halving of 50 BTC to 25 BTC is after 210000 blocks or around 3.9954 years, which is approximate at any rate based on the difficulty retargeting mechanism's best effort.

I wanted typical amounts to be in a familiar range. If you're tossing around 100000 units, it doesn't feel scarce. The brain is better able to work with small numbers from 0.01 to 1000. Which proved to be the case.

In the case it were to get really big, the decimal could move two places and cents become the new coins.

People know that the supply is capped off at 21 million. But not necessarily. At least if we're thinking in terms of market cap. If bitcoin someday manages to switch over to a completely feeless system, then sending someone 100 'satoshi' could be the equivalent of sending 100 USD.

You see, bitcoin wasn't built for small, it was built for scale. In the first version of bitcoin, I had it set to 6 decimal places but later had it changed to eight decimal places, making one bitcoin today correspond to one hundred million cent (known as satoshi today--I did not choose for it to be called this FYI). If for example, on May 7th, 2140 one 'satoshi' were to reach parity with one US dollar, then the true market cap bitcoin could reach would in fact be two quadrillion, one hundred trillion or two thousand trillion, one hundred thousand billion depending on which shorthand you prefer.

That will likely never happen. Who knows...

(Here is where I want to upend expectations, as I know 'educated guess' is not what most people expected to hear)

I also added GetBlocksToMaturity during this time. I set it at **+20**, *intentionally*. Although you can technically get to it before it reaches 120, until everyone in the network has at least 100 confirmed confirmations from the mined block, it cannot be spent. Other nodes recognize it as spendable, but the client will not. In order to compensate for any delays in the network. Yes after 100 confirmations the mined block has moved out of the immature state by then, but the client will wait until 120 blocks before informing you that the generation has matured. The figure was fixed, from the beginning. Prebitcoin, this did not exist.

Now, the fun part.

Everyone knows the **block reward**. It is represented today as **50 BTC**, or 50 * 100000000 (which represents one COIN). This gets halved every four years. What people don't know is that this wasn't always the case. It was initially going to be just **10000** (represented as one COIN).

That's right. 1/10000 or 0.0001.

Bitcoin would be *very* different today were it not for a thing called *pre-alpha* testing.

Another minute detail that has gained weight.

A matter of public discourse, more so today, is the **difficulty**. ("What was the thinking behind this?" is the question that often comes up) Everyone knows that the difficulty for bitcoin adjusts every 2016 blocks, but in truth, this wasn't and isn't the case. I will explain.

Bitcoin seems to retarget difficulty based on the time spent mining the last 2016 blocks right?

You divide the target timespan by the targetspacing.

const unsigned int nTargetTimespan = 14 * 24 * 60 * 60; // two weeks

const unsigned int nTargetSpacing = 10 * 60;

const unsigned int nInterval = nTargetTimespan / nTargetSpacing;

This gives you, 2016.

Except, in truth, it really looks at the time spent mining the last *2015* blocks to adjust the difficulty (nInterval-1).

So not only is the current difficulty not exactly 2016, but it always wasn't the case.

This was intentional.

Prebitcoin, the difficulty would adjust every, **2880** blocks. Instead of going back 14 days/two weeks, it would go back **30 days**. It also did not amount to 10 minutes per block but rather, **15** minutes. And unlike the latter, nInterval-1 was not a part of it. Now, the question of *why* is something that will be covered in the book.

In total, bitcoin had about **150** changes made to it before release (approximate at best, list of changes can be listed in the book if people want them).

These are just some of the most brought up questions I can recall.

It's hard to stitch together things that happened 10 years ago. Imagine trying to recall what you said to someone in passing, ten years ago, during a conversation, and that is what it feels like to me except I had alot more time to think about these things.

Something that people always commented on is that I seemed to have an answer for *everything*. The truth was, I had been thinking about these things for *years*, and so I knew nearly every question that someone could ask because I had likely already thought about it beforehand.

But it doesn't mean I was able to *execute* on everything. For example, one thing that I never got to work on but which was built into bitcoin was the use of safe contracts.

Safe contracts can be executed without using trust:

Tx 1 from User pays to a script that requires the signature of both the Company executing the contract and User to spend.

Tx 2 (the contract) spends Tx 1 and pays it to User. nLockTime is the time to release the money.

My thoughts for smart contracts employing an escrow system was as follows:

- Company gives user a pubkey to use in creating Tx 1.
- User privately creates Tx 1, does not broadcast it yet.
- User gives the hash of Tx 1 to Company.
- Company signs its part of Tx 2, with nLockTime set, and gives it to user.
- User broadcasts Tx 1.
- User signs his half of Tx 2 and broadcasts it.

With these steps, the user already has the Company's signed half of Tx 2 in hand before they broadcasts Tx 1, so they are assured of what bargain they are signing the money to.

This is the general pattern for safely signing contracts. Tx 2 is prepared first so the parties know what they're paying into. Tx 1 is broadcast to lock up the money and assign it to Tx 2. In other words, all parties assign their money to a pool that is controlled by the unanimous agreement of the group, but first the group has already signed agreement for the default action to take with the money, or partially signed multiple available options that a party can complete by adding the last signature.

By mutual agreement, the parties can always write another version of Tx 2 that releases the money immediately.

Bitcoin started as most things do, as a proof of concept. In the beginning, there was no precedent and I had nothing to model it after, nothing that came close to it at least. Plenty of ideas of how to execute it had existed though. To say that there wasn't anyone thinking about something like bitcoin would be naïve. That is to say, plenty of individuals had *thought* about the possibility of something like bitcoin, but none had yet put it into practice.

In order for bitcoin to come into existence it required 3 fundamental areas of knowledge converging together. In the end, what it took was a combination of math, cryptography, and a good working knowledge of C++ to make it work.

Certain aspects, specifically the POW and the hash function had already existed and were by then taken as a matter of implementation. Other things, in hindsight, were created by pure coincidence. The blockchain for example. The blockchain came to be a way to solve all the issues that a decentralized currency could encounter.

To put this into effect, I give the example of the The Byzantine Generals' Problem

A number of Byzantine Generals each have a computer and want to attack the King's wi-fi by brute forcing the password, which they've learned is a certain number of characters in length. Once they

stimulate the network to generate a packet, they must crack the password within a limited time to break in and erase the logs, lest they be discovered. They only have enough CPU power to crack it fast enough if a majority of them attack at the same time.

They don't particularly care when the attack will be, just that they agree. It has been decided that anyone who feels like it will announce an attack time, which we'll call the "plan", and whatever plan is heard first will be the official plan. The problem is that the network is not instantaneous, and if two generals announce different plans at close to the same time, some may hear one first and others hear the other first.

They use a proof-of-work chain to solve the problem. Once each general receives whatever plan he hears first, he sets his computer to solve a difficult hash-based proof-of-work problem that includes the plan in its hash. The proof-of-work is difficult enough that with all of them working at once, it's expected to take 10 minutes before one of them finds a solution and broadcasts it to the network. Once received, everyone adjusts the hash in their proof-of-work computation to include the first solution, so that when they find the next proof-of-work, it chains after the first one. If anyone was working on a different plan, they switch to this one, because its proof-of-work chain is now longer.

After about two hours, the plan should be hashed by a chain of 12 proofs-of-work. Every general, just by verifying the difficulty of the proof-of-work chain, can estimate how much parallel CPU power per hour was expended on it and see that it must have required the majority of the computers to produce in the allotted time. At the least, most of them had to have seen the plan, since the proof-of-work is proof that they worked on it. If the CPU power exhibited by the proof-of-work is sufficient to crack the password, they can safely attack at the agreed time.

This is an example of what the proof of work could do, it simultaneously converges the general view and generates a computational proof of the majority consensus without having to trust anyone.

(*Skipping a few chapters here*)

In the beginning, I made it ridiculously easy for anyone to mine bitcoins (difficulty was downright negligible for the first year), so as to spur adoption, but for the majority of the first year, it was just me and Hal who seemed to be using it. Most of the time, it was mostly my own PCs that were connected to the network.

Still, it was exciting to see it working. This too allowed some leeway, as I was the only developer at the time, bug fixes affected less users and made it easier to correct them.

The premise for bitcoin in the beginning was simple--it was a new electronic cash system that used a peer-to-peer network to prevent double-spending. It was completely decentralized with no server or trusted parties.

The main properties as it were for bitcoin at the time of release were:

- Double-spending is prevented with a peer-to-peer network.

- No mint or other trusted parties.

- Participants can be anonymous.

Later on this was expanded to cover things like:

- Transfer money easily through the internet, without having to trust third parties.
- Third parties can't prevent or control your transactions.
- Be safe from the instability caused by fractional reserve banking and bad policies of central banks. The limited inflation of the Bitcoin system's money supply is distributed evenly (by CPU power) throughout the network, not monopolized by the banks.

Bitcoin had to be created with a focus on general consensus. But without a trusted third party. I struggled with this notion. The concept in *practice* was foreign. How can I get all these people to agree on something, and trust that the network itself was not being modified or tampered with indiscriminately?

The answer was a protocol. The model itself, which I thought none had existed, came into view very quickly. It had been staring at me all this time, I had been using it for years, and had never taken into account how efficiently it worked. Sometimes things work so well that you never question their existence. They just are. The answer, was the internet itself. And the network, was peer-to-peer.

To understand what I mean, you really have to trace the history of how the internet as we know it came to be. Specifically the role TCP/IP, HTTP and SMTP protocols played in the formation of it. Very early on, there was a debate as to which protocol would win, as there was more than one.

TCP or what it later came to be was spun off out of research conducted at the Defense Advanced Research Projects Agency (DARPA) in the 1960s. This was a follow-up to the next generation protocol for the ARPANET. This program evolved into the protocol we know today as the Transmission Control Program.

It grew so large that they called for a split of protocol layers in order to designate divisions of functionality. The reason for being of the network was distilled down to one purpose, that it should only provide the function of efficiently transmitting and routing traffic between nodes. And this, is how TCP came to be. Using this end to end design, it finally became possible to connect almost any network to the ARPANET. A router provided an interface to each network. All of this finalized into one standard protocol that in 1974 resulted in the first TCP specification.

Four versions of this TCP protocol were developed, but only the last one, TCP/IP won out. Eventually ARPANET was migrated to TCP/IP in 1983 with the DoD declaring it the standard protocol for all military communications. This led to a variety of TCP/IP stacks being offered up for DOS and Windows, but it wasn't until 1995 that Microsoft made the bold move of including their own native TCP/IP stack in Windows 95, and with that the internet was born…

(I want to skip ahead and dedicate sometime now to how I set up the website as this is something that required a bit of preparation on my end, not the least of which required some delicate maneuveuring.)

Registering a website anonymously is not an easy task. Usually registrars ask for personal information, something I was not willing to give, and ownership is usually tied to a CC which links you directly.

I knew that in order for this to work, I would have to do it anonymously. All the way through. And I knew then as I know now, that after my work was done, I would leave the community entirely.

(I want to take a moment here to clarify something. Some may seem to refer to my working methods as paranoid, but in truth, with all the surveillance, who was more right in the end? That I was merely paranoid for no obvious reason is just not true. Cautious is the key word. In any case, suspicions were confirmed only two years later when I left.)

The first thing I did was register the website anonymously. At the time there was the service anonymous speech that let you do just that. The difference was that since it was registered in Japan, it was not subject to foreign state requests about their subscribers nor the contents of the emails sent or received. This was appealing me to because I wanted to ensure that there were a few layers of protection between me and any malicious would be cypherpunks, which is why I married this with TOR and consistently used second hand, unregistered equipment. I never registered anything to my name, and like the website, I paid in other methods, be it cash, prepaid cards, etc.

There were other claimed anonymous services at the time, but most of these weren't anonymous at all and wouldn't protect you from a subpoena (a court order) or even a letter from a lawyer.

For one thing, I used separate emails in order to be certain that if one were compromised, I still had the others to rely on. During this time I used their anonymous email service, along with TOR which reassigned my location as appearing on the west coast. To render confusion. I purchased the first domain using anonymous speech which as I mentioned before, allowed you to pay by other methods, such as cash in the mail.

In preparing for the release of bitcoin, I had to do various steps. I had to time everything perfectly.

One of the first two people I emailed about bitcoin, or rather what would come to be known as bitcoin as I had not yet decided on a name as it was still by then *electronic cash* at that point, was Adam Back and Wei Dai.

(*I don't see how this could come as a surprise to anyone as electronic cash is still, literally, in the title to this day. I should point out that the paper was written (last minute) after the software had been finished, as it should be, unlike the norm today.*)

I emailed Adam first, who then pointed me to Wei, both of which I included in my original design paper. I shared with them the prerelease draft of the design paper, but I suspect neither of them have it any longer (actually I know this for a fact--if you don't believe, then ask), and the only paper that is available today is the one I sent to the mailing list, which was after the fact. When I refer to the design paper, I refer to what is known today as the **whitepaper**. I renamed it, upon public release.

Adam Back had been a regular on the mailing lists, and one of the few people thinking about how someone could come up with a true untraceable ecash around the same time as me. He was a big proponent of distributed ecash, something that many people had tried to do, but none had done successfully. For Adam, I believe his concern was how do you stop it from hyper inflating at the rate of Moore's law, which until Bitcoin, no one had found a way to do this. Most of the early ideas proposed in the cypherpunks community either remained contained to academic papers, or had been deployed but never found traction and didn't work for various reasons (a notable example is Digicash).

I had learned about Adam through his posts on the mailing list, which is how I also found out about hashcash. He had written about it on the cypherpunk mailing list detailing its properties, and people were comparing it to digital gold and how it could be implemented for a distributed ecash.

Although Adam was the one responsible for Hashcash, and although it had been used up to that point mostly for Anti-Spam, there were various other utilities for it where it could be used. One of them was as part of the mining function for bitcoin with some key differences. Hashcash worked on whole bits, bitcoin mining was extended to allow fractional bits, 2^32 being the original and starting difficulty in bitcoin. This was because namely a double hash, and switching from SHA1 to SHA256 and bitcoin defining the work of finding hashes as < 2^x where x is the log2(difficulty)+32 in bits. Although the difficulty of bitcoin is expressed in exponents, it could also be expressed in bits, it is mostly by convention as the number was smaller and more readable.

When I did decide to contact him and ask if I could cite his paper, I had already built and designed a prototype, and it had already been using hashcash as the mining function. In this moment, he pointed out the similarities in my paper and recommended I should look into Wei Dai's b-money.

When I reached out to Wei, I had already studied his b-money page and sought to include it in my paper. Having others see what formative ideas helped shape the creation of bitcoin would I hope, help them better understand my motivations. In his essay, Wei proposed the use of a proof of work as a means to creating money. I would assume that Wei's proposal came in a response to Adam Back's Hashcash, which had been posted on the same cypherpunk mailing list a year earlier. In the paper, Dai outlined the basic elements that all modern cryptocurrencies should possess, properties that included:

- A specified amount of computational work through, in this case PoW
- That work is verified by the community who update the global ledger or "blockchain"
- That the worker is given funds for their best effort, in this case the user
- That the exchange of funds is achieved by way of cooperative recordkeeping and authenticated through the use of cryptographic hashes
- Contracts are compulsory through nodes broadcasting and signing transactions with digital signatures using public key cryptography

Dai described the virtues of cryptocurrency quite clearly, so that as early as 1998, the basic building blocks for a currency like bitcoin had already existed. And indeed, many different iterations had to fail in order to pave the way for bitcoin and all subsequent cryptocurrencies. But Wei provided a basis for which I could take existing ideas and take it a step further by fulfilling his vision of a fully decentralized currency.

Some might wonder why I went to the extent of getting their work cited in my paper when I had already for the most part launched and built the first test network of bitcoin, and the truth is although

everything had already been ironed out and was running, my concern came to whether the community would accept or reject my work that I had spent the better part of two years working on. In academia, as in daily life sometimes you need the backing of higher ups in order to get your work through the door.

I am quite certain that had I just gone ahead and announced on the mailing list my work (and intentions) without any citations or references, it would have likely been dismissed as another failed attempt at a decentralized currency. But here then were two well-known (and respected) individuals in the cryptography community, who were in some sense, vouching for me by having their names included in the paper. This was my reasoning, so that when it came time to announce, there was already some backing or footing to get it through the door and into the hands of the people that would be the harshest to criticize it.

When I left the community (skipping a few, well okay many chapters here for the excerpt), I transferred ownership of the domain to Martti, and left the repository in the charge of Gavin. In April I decided that now should be the time to leave (in truth I had already left the community much earlier by Christmas the year before but decided to officially make my departure known--and if you don't believe, ask the core developers of the time, like Mike), because maintenance and development had been taken over by a skilled team of developers, so they didn't really need me anymore, and I really had no desire to continue. I sent my final correspondence to the core developers, and designated the responsibilities to others. I corresponded for a little bit more, and soon after erased everything that had linked me to Satoshi. I tried my best to leave no affirmation that I had ever existed at all, and everything that did link me I put into a series of files, so in the case I left, anyone could take on the persona.

There are other, more serious reasons for leaving that I will not include here but will be mentioned in the book.

Rewinding time here.

I had already been thinking about ideas for a currency starting from around mid 2006, but it wasn't until a year later that I really started figuring out how to make a prototype out of those ideas. My understanding of cryptography was not particularly advanced, nonetheless I took what I knew and what existed and what was already working and made the improvements where I saw fit. I studied the current research material coming out at the time and implemented what I thought would be long enduring and long withstanding. Some of the protocols and standards were just coming into compliance and I nonetheless decided to include them for the sake of future proofing.

I know it is easy to imagine that bitcoin came out of thin air but in reality, that was not the case. It arose out of the many failed attempts by many *groups*, and the only reason it succeeded was because it was at the right place, at the right time. And although I did not *exactly* time it for the beginning of a financial crisis—who could predict that (curious the answer though--wait for the rest), it did serve as an impetus and I think had it not been for that, adoption would not have taken off as it did. Other ideas--using cryptographic algorithms, proof of work, timestamps, had already existed. But no one had tied all these things together in a way that mirrored the form in which the internet was structured. I read every academic paper of the last fifteen years before I dived into it and remarked how in every failed attempt

at making something like bitcoin work, one key ingredient was missing. All of them had bits and pieces of it, but not the complete image.

Peer to peer was the underlying aspect to bitcoin that made it superior to all other previous attempts. Its advantage was the network architecture that off-loaded the workload between peers, forming a network of nodes. Every node contributed to this network, without the need for a central authority or third party, and without any servers. Peer to peer itself was born out of a need to decentralize, while ARPANET allowed anyone to request and serve content, it strained beyond offering anything else beyond simple routing. USENET, a system designed in 1979 that was a precursor of sorts to BBS systems, allowed for a decentralized and distributed system. But what set this apart from other BBS or web forums of the time was that USENET was distributed, with the absence of a central server and dedicated administrator. USENET instead, was decentralized.

This one key difference is what led to technologies like P2P file sharing, and services like Napster, and subsequently Bittorent which spun out of Mnet. Mnet itself was a peer to peer file sharing service that employed a digital currency called Mojo, which in a fully distributed network provided an incentive against attacks, one of the first "smart contracts."

I mention USENET briefly because when I announced bitcoin, certain things that were brought to my attention only after the fact, such as Public Time Stamp were unknown to me before release. The internet was so vast that for all intents and purposes, even I did not know everything.

Dustin Trammell is one of those that brought that insightful realization to my attention, for although I thought I had covered every base, there were still things left to learn. When he emailed me the day I publicly announced bitcoin, he shared with me a surprising use of timestamps that did not employ USENET. This is why later on I made a point of including other existing ideas that I may have neglected, such as Bit Gold as a forerunner to bitcoin.

Still, it was grounding to see that there were things I hadn’t yet discovered. Dustin continued to correspond with me in those early days, making suggestions, which in turned helped me make bitcoin more presentable, less clunky and a lot more functional. Throughout the next several days we discussed ways in which bitcoin could catch on with the public. What I liked about Dustin was his openness, for both of us shared a deep interest in the idea of money and cryptography. People assume that I was not a "cryptographer", and in the truest sense of the word I wasn’t, but it did not signify that I did not know how to deploy cryptographic *proofs* into what evolved into bitcoin.

Justin was also one of the few people providing me feedback on the first (what I considered) stable version of bitcoin, v.0.1.3. Prior to this, the software was bug ridden and I was ironing out bugs in the software on an hourly basis the first few days.

One of the more interesting questions he called upon on is what would happen in the case that there is a single node with the most drawn CPU power, where if one is more powerful than the rest, it is assumed that it will always win the majority of coins. I explained to him in the simplest way I could and as I had always done in the past, by using an analogy. Just as I had tried to do with the Byzantine

General's problem, I felt that the best way of explaining things to new users was to take an existing idea or concept that was easy to understand, and apply it to a likely (feasible) scenario, in this case it was relative to performance.

I used for starters the analogy of a fast car, as I could tell he was thinking in terms of horsepower. I explained to him to think of it not as a race where if one car is twice as fast, (it is assumed) it'll always win. Rather it's an SHA-256 that takes less than a microsecond, and each individual guess has an equal but independent chance of success. Each computers chance of finding a hash collision is linearly proportional to its CPU power. So although a computer may be half as fast, it would simply mean it would get half as many coins, rather than none. Bitcoin as it was, was not a winner take all system.

The thing that people should understand about bitcoin at its core, is that it along with all timestamp servers shared the basic functionality of periodically collecting things into blocks and hashing them into a chain.

Bitcoin was also at its most vulnerable in the beginning when the total network CPU power was small. But this was offset by the fact that the incentive to attack it in the beginning was also small.

I think people wonder why it took so long for something like bitcoin to reach critical mass, but the truth is that the technology to make bitcoin happen did not fully exist yet, at least not in maturation and although there were a lot more people interested in the 90's, after more than a decade of failed trusted third party based systems (Digicash, etc), they saw it as a lost cause. My fear was that they ("they" being the cryptography community) would not make the distinction. But bitcoin was so radically different from all other previous attempts that I made it a point for them to make that distinction. That they would realize this was the first time I knew of that a completely non-trust based system had been proposed.

I was so concerned with what the cryptography community would think that it resulted in me disregarding mentioning any failed attempts by referencing Digicash in the original design paper as a forerunner to bitcoin for fear that people would dismiss it before it got started, and move on. So instead I referenced the good virtues of ideas like b-money and the hashback proof of work system, which itself was inspired by the work of others, which Adam references in his paper. Taking the good ideas, and omitting the failed ones was the only way I saw this succeeding in the eyes of such harsh skeptics (maybe critics is the better word). Digicash otherwise would have been cited, had it been a non-trust based system.

(*Hal would've called them cryptographic graybeards. He always had a sense of humor*.)

Hal alluded to this that by 2019, as I write this that Bitcoin and its chance of success would be a long-odds investment, in other words, not very likely. It would either work really well and catch on, or fail spectacularly. And although I saw many uses for bitcoin outside of the traditional ones, such as narrow niches where traditional currency didn't really fit in--reward points, donation tokens, in game currency, things like that, I never did expect bitcoin to catch on the way it did and to be in direct competition with fiat currency. But that was exactly what happened. It went beyond expectations.

Bitcoin for all intents and purposes was built very secure with few points of failure and built mostly for the long term, nevertheless some questioned the methods to which I arrived at where I did.

Many people may wonder why bitcoin was designed in C++ at its core, and the answer mostly had to do with memory. Not withholding resiliency and adaptability. I was most concerned with reliability and attacks when thinking of what programming language to use, attacks on the network being the prime concern.

I group attacks into two classes:

1) Attacks that can only be done by someone actually in the chain of communication

2) Attacks that can be done by anyone on the Internet from anywhere

The first method of attack exposes you to people in your house or company on your local LAN, admins at ISPs in between, and the LAN on the recipient's side. The second exposes you to a billion people who can self-select to be attackers and get economy of scale when they develop one technique to attack multiple victims.

The second method of attack is the reason why bitcoin came to be designed in C++. You see, bitcoin had to be resilient to all forms of attacks, as it had a large attack perimeter. As it was open to the internet, it had to carry a strong requirement for consistency. C++ provided that. It meant a firm control over memory usage, which meant that a security critical application like bitcoin that is directly exposed to the internet could still provide reliability to local clients while communicating with a large number of untrusted endpoints. This required keeping a firm control over resources like memory, which C++ made a good job of. Unlike other programming languages, C++ offered consistent control over memory usage while also optimizing for speed and performance. What were the other reasons in choosing C++? Not mentioning the above, C++ had a strong base and foundation, the technology was rock solid and new features were being added all the time. It was an obvious choice.

I wrote this to him by email explaining further:

Sending by IP requests a new public key, so yes, it's vulnerable to type 1 man-in-the-middle. If that's a concern, sending to a Bitcoin address doesn't have that vulnerability, although there's a small privacy tradeoff. I have a feeling most of the time people will get Bitcoin addresses off of non-SSL websites and unsigned cleartext e-mail, which is already vulnerable to type 1 and type 2 through DNS poisoning.

One solution would be to use both the IP and Bitcoin addresses when sending (maybe 1.2.3.4-1Kn8iojk...), where the recipient uses the public key of the Bitcoin address to sign the new public key to prove that you're sending to who you think you are. If the system starts to be used for real business purposes, I will certainly implement that. Another solution is to use SSL.

For now, it's pretty obvious that if you send to an IP, you didn't give any other identifying information about the recipient, so you're blindly sending to whoever answers that IP.

Another feature for later is an option to encrypt your wallet.

This is now where I want to dedicate a moment to an individual who should be credited with much of the initial success, for their help was instrumental. I take little if any responsibility, for this person is really the one that should be thanked.

(*I want to introduce you Hal, someone whom I considered a friend, and the first believer in what I was trying to do. Hal, if you're up there mate, I hope you're doing well, you should know this excerpt is really called "Bitcoin, me, and Hal Finney". God speed. Save me a seat at the table*)

Harold Finney (Hal for short) was one of the most intellectually bright minds I ever had the chance of parlaying words with. Not only was he the person responsible for the first reusable proof of work system, but he was genuinely skilled. He also was involved in the continued development of PGP (For the record, I've never actually had to use PGP for anything, and doubt I will this time either). I held a tremendous amount of respect for him, not the least of which was his dedication to ensuring that bitcoin would survive, because even when others had shown little interest in bitcoin early on, Hal was the only one who kept the network going alongside myself.

I had known about Hal for some time from the cypherpunk mailing list, and so I respected his thought process, as he was someone I held to high esteem.

Hal was the first person to build on the proof of work concept, and the first to come up with a reusable proof of work. This was essential in prototyping bitcoin, for his work on RPOW as *e-money* is what inspired me to use the POW in the first place. Much to the chagrin of people today, I still believe in the virtues of a Proof of Work system (I will explain why in the books). Hal ensured that the value of an RPOW token was correspondent to the value of the real world resources required to mint the POW token. In his case, the POW token was a piece of Hashcash, which was developed off the work of Adam Back.

Although his RPOW never saw significant traction and no real use, it was like bitcoin, based on the Hashcash POW. This meant that both had come from the same origins, but both had achieved difference outcomes. Whereas the RPOW Hal devised was based on the Hashcash POW, it relied instead on the hardware trusted computing function, meaning the computer will consistently behave in an expected manner and those behaviors are enforced by the computer hardware whereas Bitcoin relied on a decentralized P2P protocol. Bitcoin in this sense was better protected and could be better trusted, because while RPOW was protected by the private keys stored in the TPM (Trusted Platform Module) hardware, anyone holding those keys could be capable of undermining that trust.

The difference was bitcoins instead were mined using the Hashcash POW function by individual miners and verified by the nodes themselves in a P2P network.

Hal was also the first person I ever sent coins to, being my first recipient and the first official bug reporter, i.e. the first person to report on a bug.

My conversations with Hal in the beginning consisted mostly around debugging the first version of the software, v0.1.0 ALPHA, and although I personally hadn't experienced any instances of hiccup in using the software, I knew that opening it up to more users would help me both identify and fix issues that I could not experience or replicate solely on my end.

Bitcoin was a security intensive application, not the least of which because it was internet facing. In order for it to work, all bugs had to be ironed out from the software before release, or as many as one could find. Debugging consists of searching for a "bug" in the software until you find where the failure occurs and then determining what the incorrect parameters or code are in order to make the necessary changes.

GDB reads only enough symbol data to know where to find the rest when called upon, while GCC avoids producing debug symbol output for types that are not used in the source file that is being compiled.

In the case an application works in a debug build, but fails in the release build, it’s likely that the compiler optimizations are the cause and the culprit is a defect in the source code. In a debug build, optimizations are turned on and debug symbols are not emitted. So in order to isolate the bug, you have to disable the compiler optimizations until you can locate the problem file. In a gist, this is how debugging works.

In Hals case, no symbols came up. This was because I had stripped them out as it would have increased the executable file size nearly 45 mb. At the time I couldn't justify it because I had not encountered any outright exception, but afterwards I realized it was a mistake as the software still had a lot of bugs.

At the time, Hal was using MSVC (Visual Studio for those that aren't aware), and experiencing crashes in launching bitcoin. Running cygwin, a UNIX like command interface, which allows you to launch windows specific applications on it, in this case the application being bitcoin, Hal wasn't experiencing a crash.

This led me to reproducing the bug on my end and isolating it to "mapAddresses.count" which wasn't essential to the software, and so I was willing to just delete that part and turn off optimization until I could figure out what was going on because I wasn't able to reproduce it any longer on my end.

This (seemingly) small issue was troubling me so much that I proceeded to spend the next few hours trying to stress test the software and recreate the issue. Three hours later, I finally figured out what was the problem. The issue was that when I started a thread and did mapAddresses.count, it was calling upon a memory access violation, or segfault. This usually happens when a program attempts to access a memory location that it's not allowed to.

That a single snippet of code I added for last and didn’t even really need had made a mess of the release really caused me distress and frustrated me, so I settled with putting out a new version, v.0.1.1 that omitted the mapAddresses.count. Once I deleted that single instance and line of code and turned off optimizations, everything was running fine. It went from chaos in the morning to “everything is fine” by the end of that day.

We continued to work on version v0.1.2 which I sent to him to test out, and the software was still having problems, with his node becoming unresponsive which I attributed to either ThreadSocketHandler or ThreadMessageHandler. I sent him over a debug build to find the cause and narrowed it down to a select failed error, which determines the status of one or more sockets, specifically "select failed: 10038". I was having such a rough day of it, feeling defeated that I told Hal that the Internet is a brutal, rough and tumble place. Eventually I found a fix, the select error itself not being a major issue, but it did lead the communications thread to getting blocked on a socket. Had I not fixed this error, node's communications would be intermittent and sometimes go dead. Connections are still established, but no data passes through.

That was why any generated blocks were not being accepted since he couldn't broadcast them, forcing the other nodes to leave the branch behind. Also why he couldn't generate, as it didn't appear as if he was connected. By the time I sent him v0.1.3, everything was for sure, this time running fine. As a token of my appreciation, I sent Hal some coins so he could play around with it. In a span of a few days, I had gone from being completely sure that the first version would be mostly bug free to all hell broke loose in a matter of minutes and I am spending the entire time trying to find out ways to fix it and get it working.

Hal wanted to send me some of his coins which were about to mature, unfortunately (as I had Port 8333 on my end closed) I couldn't receive any incoming connections from where I was, which made things difficult.

(I want to take a moment here to explain that *prebitcoin*, this was not the port I chose, instead it was Port **2222**. The way that TCP ports work is the IANA (Internet Assigned Numbers Authority) is the entity to assign port numbers for specific uses. In other words, Port 8333 just kind of became the unofficial assigned port for Bitcoin by convention--it is not and was never registered with the IANA.)

So instead I told him to send it to my address, which meant I would receive the payment the next time I was online. Sending payments to an address consisted of sending it to a hash of their public key, so that the next time a user connects, they see the payment. This had the disadvantage of no comment information being sent, and if you want to maintain anonymity, the address should only be used one time. This is useful though when the recipient isn't online, if they are it's simply a matter of entering their IP address, getting a new public key and sending the transaction with comments.

That night, v0.1.3 on Hals computer crashed, *yet again*. It seemed like I would be settled to having to fix errors non-stop. I sent him the debug build for it and we both came to the conclusion that the disk full that was the culprit. I had always disliked projects that consisted of big dependencies but there was just no avoiding it as each one is essential. I suggested to Hal that if he did build the dependencies, to let me know how it went. I was perplexed though as to how he got a read exception rather than a write exception upon his disk filling up. I assumed this might have corrupted his block file, and suggested that he delete them outright if the issue happens again. Deleting the block files wouldn't permanently delete them as it would re download the block chain. Even without backing up the whole directory including the database subdirectory, the most crucial file to download is the wallet.dat, where the private key information is stored.

The database had the unfortunate case of naming its files "log.0000000001" which meant that to those familiar with databases, it would mean delete-and-lose-everything-in-your-other-files. I tried to put them out of harm's way by putting them in the database subdirectory. Later on I wanted to avoid this by writing code that would flush the logs after every wallet change so wallet.dat would be safe.

Most don't know but it was Hal who really kept bitcoin going the first few days, and what with his one sole node receiving incoming connections which was the main if not sole thing keeping the network going the first day or two. Was it not for Hal, I don't know what I would have done in those first few days.

To this day, I still think about how good of a person Hal was and how if it wasn't for him, bitcoin would have not succeeded the way it did. When I had no support, when it was just me, Hal was the only one other person who believed in what I was trying to do. If any one person should receive credit for bitcoin and its initial success, it is him. Hal sadly passed away in August 2014 to a debilitating illness, ALS. But until the very end, Hal kept up the fight and was as happy as one could be despite the odds. He was aware of his own mortality and accepted the time he had left as a virtue.

That famous quote of his that often gets brought up where he explains how he got interested in cryptography emphasizes the similar beliefs we all shared. The reason that united us all.

"It seemed so obvious to me. Here we are faced with the problems of loss of privacy, creeping computerization, massive databases, more centralization - and Chaum offers a completely different direction to go in, one which puts power into the hands of individuals rather than governments and corporations. The computer can be used as a tool to liberate and protect people, rather than to control them."

— *Finney, Cypherpunks Mailing List in 1992*

Hal was smart enough to guess what the future of cryptocurrency could look like, guessing rightly that eventually most people would be aware of what it was--that it wouldn't remain a niche thing for long. And uniformly, 10 years ago, I sort of came to the same conclusion, that we too would be using digital currency in some form as many others had imagined it could be used. What I didn't anticipate was just how fast, or that bitcoin would be the one to spur it.

And with that, is where this magical story into the history of what once was, concludes..