Subject: **Questions about BitCoin**

From: **Mike Hearn** <mike@plan99.net>
Date: Sun, Apr 12, 2009 at 12:46 PM
To: satoshin@gmx.com

Hi Satoshi,

I read your paper on BitCoin with great interest. I found it a bit confusing though - I believe it may be easier to follow if you provide some examples.

Specifically, it's not quite clear to me what blocks contain. If I understand correctly, there is only one (or maybe a few) global chain[s] into which all transactions are hashed. If there is only one chain recording "the story of the economy" so to speak, how does this scale? In an imaginary planet-wide deployment there would be millions of even billions of transactions per hour being hashed into the chain. I realize that each PoW can wrap many transactions in one block, nonetheless, that's a large amount of data to hash. If there are many chains, how are transactions assigned to each chain such that it is still difficult to overpower the network? Eg, if there are 10 global chains, the amount of cpu power you need to beat the system is only 10% of what it was previously.

I also wonder if the assumption of 1 core = 1 vote is sound. If the majority of nodes are on standard computers, it seems likely that an attacker could use FPGA or custom ASICs to get significantly better performance. What are your thoughts on using custom hardware to beat the chain?

I found the section on incentives hard to follow. In particular, I'm not clear on what triggers the transition from minting new coins as a reason to run a node, to charging transaction fees (isn't the point of BitCoin largely to zero transaction costs anyway?). Presumably there's some human in charge of the system - eg, you decided somehow that 24 million coins was a good number to have, and would distribute some kind of rules file saying "coins minted after this timestamp must have an N+1 zero bits prefix", which honest nodes enforce.

How did you decide on the inflation schedule for v1? Where did 24 million coins come from? What denominations are these coins? You mention a way to combine and split value but I'm not clear on how this works. For instance are bitcoins always denominated by an integer or can you have fractional bitcoins?

So many questions :) But it's rare that I encounter truly revolutionary ideas. The last time I was this excited about a new monetary scheme was when I discovered Ripple. If you have any thoughts on Ripple, I'd also love to hear them.

thanks -mike

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Sun, Apr 12, 2009 at 10:44 PM
To: Mike Hearn <mike@plan99.net>

Hi Mike,

I'm glad to answer any questions you have. If I get time, I ought to write a FAQ to supplement the paper.

There is only one global chain.

The existing Visa credit card network processes about 15 million Internet purchases per day worldwide. Bitcoin can already scale much larger than that with existing hardware for a fraction of the cost. It never really hits a scale ceiling. If you're interested, I can go over the ways it would cope with extreme size.

By Moore's Law, we can expect hardware speed to be 10 times faster in 5 years and 100 times faster in 10. Even if Bitcoin grows at crazy adoption rates, I think computer speeds will stay ahead of the number of transactions.

I don't anticipate that fees will be needed anytime soon, but if it becomes too burdensome to run a node, it is possible to run a node that only processes transactions that include a transaction fee. The owner of the node would decide the minimum fee they'll accept. Right now, such a node would get nothing, because nobody includes a fee, but if enough nodes did that, then users would get faster acceptance if they include a fee, or slower if they don't. The fee the market would settle on should be minimal. If a node requires a higher fee, that node would be passing up all transactions with lower fees. It could do more volume and probably make more money by processing as many paying transactions as it can. The transition is not controlled by some human in charge of the system though, just individuals reacting on their own to market forces.

Eventually, most nodes may be run by specialists with multiple GPU cards. For now, it's nice that anyone with a PC can play without worrying about what video card they have, and hopefully it'll stay that way for a while. More computers are shipping with fairly decent GPUs these days, so maybe later we'll transition to that.

A key aspect of Bitcoin is that the security of the network grows as the size of the network and the amount of value that needs to be protected grows. The down side is that it's vulnerable at the beginning when it's small, although the value that could be stolen should always be smaller than the amount of effort required to steal it. If someone has other motives to prove a point, they'll just be proving a point I already concede.

My choice for the number of coins and distribution schedule was an educated guess. It was a difficult choice, because once the network is going it's locked in and we're stuck with it. I wanted to pick something that would make prices similar to existing currencies, but without knowing the future, that's very hard. I ended up picking something in the middle. If Bitcoin remains a small niche, it'll be worth less per unit than existing currencies. If you imagine it being used for some fraction of world commerce, then there's only going to be 21 million coins for the whole world, so it would be worth much more per unit. Values are 64-bit integers with 8 decimal places, so 1 coin is represented internally as 100000000. There's plenty of granularity if typical prices become small. For example, if

0.001 is worth 1 Euro, then it might be easier to change where the decimal point is displayed, so if you had 1 Bitcoin it's now displayed as 1000, and 0.001 is displayed as 1.

Ripple is interesting in that it's the only other system that does something with trust besides concentrate it into a central server.

Satoshi

From: **Mike Hearn** <mike@plan99.net>
Date: Mon, Apr 13, 2009 at 1:39 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

Thanks Satoshi,

I tried the app yesterday. It seems to work pretty well running on Wine (I tried it on MacOS but it should run on Linux too, and will try that next week when I am back at work).

In the lower right hand corner it has a block count which increases rapidly and then stops. Is this the length of the global chain? It seems to advance far too fast for that. Or is this the number of genesis blocks that have been tried but did not result in a partial collision? I'm not sure if the way it stops and starts is expected, or some glitch caused by it running under emulation. My best guess - it is the length of the global chain, and the rapid advance at the start is as the software downloads and verifies the preceding blocks in the chain as being valid.

With regards to the buyer/seller experience, I understand that the global chain advances at about 6-7 blocks per hour under the current settings. If we assume that 0.1% is a good risk rate, then z=5 thus any transaction must wait a bit less than an hour before being solidified in the chain. As micropayments for things like web content or virtual goods are by definition something that requires low overhead, waiting an hour seems like quite a significant hurdle.

I understand that nodes attempt to find a POW to advance the global chain in an uncoordinated fashion. This sentence however:

"If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains."

is confusing for me, because it appears the only way the honest chain can grow faster than a chain worked on by 1 attacking cpu is if the keyspace to scan looking for a partial collision is sharded evenly amongst the participating honest nodes. That way the speed at which collisions are found would be proportional to the number of nodes. Yet I don't see any discussion of such work sharding, which obviously adds complexity. Likewise:

"To compensate for increasing hardware speed and varying interest

in running nodes over time,
the proof-of-work difficulty is determined by a moving average
targeting an average number of
blocks per hour. If they're generated too fast, the difficulty increases."

How is the required difficulty of each block communicated through the network and agreed upon?

Thanks once again. I have yet more questions but this is enough for one email :) I will be happy to summarize these discussions into an FAQ-like document at some point. Apologies if the questions seem trivial.

-mike

From: **Mike Hearn** <mike@plan99.net>
Date: Mon, Apr 13, 2009 at 10:51 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

Something else that isn't clear to me - does the global chain only get extended when there is actual work to do? Currently it seems to grow all the time, although there are only a few people in the network. So presumably it gets extended with null blocks. Is this actually required? The timestamping doesn't have to be actually in parallel with real time does it ... it's merely establishing an ordering of events.

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Mon, Apr 13, 2009 at 11:00 PM
To: Mike Hearn <mike@plan99.net>

Mike Hearn wrote:
My best guess - it
is the length of the global chain, and the rapid advance at the start
is as the software downloads and verifies the preceding blocks in the
chain as being valid.

Right. I'm trying to think of more clear wording for that, maybe "%d network blocks" or "%d block chain".

If we assume that 0.1% is a good risk rate, then z=5 thus
any transaction must wait a bit less than an hour before being
solidified in the chain. As micropayments for things like web content
or virtual goods are by definition something that requires low
overhead, waiting an hour seems like quite a significant hurdle.

For the actual risk, multiply the 0.1% by the probability that the buyer is an attacker with a huge

network of computers.

For micropayments, you can safely accept the payment immediately. The size of the payment is too small for the effort to steal it. Micropayments are almost always for intellectual property, where there's no physical loss to the merchant. Anyone trying to steal a micropayment would probably not be a paying customer anyway, and if they want to steal intellectual property they can use the file sharing networks.

Currently, businesses accept a certain chargeoff rate. I believe the risk with 1 or even 0 confirming blocks will be much less than the rate of chargebacks on verified credit card transactions.

The usual scam against a merchant that doesn't wait for confirming blocks would be to send a payment to a merchant, then quickly try to propagate a double-spend to the network before the merchant's copy. What the merchant can do is broadcast his transaction and then monitor the network for any double-spend copies. The thief would not be able to broadcast during the monitoring period or else the merchant's node would receive a copy. The merchant would only have to monitor for a minute or two until most of the network nodes have his version and it's too late for the thief's version to catch up and reach many nodes. With just a minute or two delay, the chance of getting away without paying could be made much too low to scam. A thief usually needs a high probability of getting an item for free to make it worthwhile. Using a lot of CPU power to do the brute force attack discussed in the paper in addition to the above scam would not increase the thief's chances very much.

Anything that grants access to something, like something that takes a while to download, access to a website, web hosting, a subscription or service, can be cancelled a few minutes later if the transaction is rejected.

is confusing for me, because it appears the only way the honest chain can grow faster than a chain worked on by 1 attacking cpu is if the keyspace to scan looking for a partial collision is sharded evenly amongst the participating honest nodes. That way the speed at which collisions are found would be proportional to the number of nodes. Yet I don't see any discussion of such work sharding, which obviously adds complexity.

The keyspace is huge, 2^256. The thing being hashed includes the node's public key and a random nonce, so the chance of any two nodes duplicating work on the same space is negligible.

How is the required difficulty of each block communicated through the network and agreed upon?

It's not communicated. The formula is hardcoded in the program and every node does the same calculation to know what difficulty is required for the next block. If someone diverged from the formula, their block would not be accepted by the majority.

Thanks once again. I have yet more questions but this is enough for one email :) I will be happy to summarize these discussions into an FAQ-like document at some point. Apologies if the questions seem trivial.

No problem, thanks for testing it on Mac Wine.

Satoshi

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Mon, Apr 13, 2009 at 11:11 PM
To: Mike Hearn <mike@plan99.net>

It keeps getting extended all the time. If it stopped, an attacker would have time to catch up. Don't worry, empty blocks aren't very big.

As you say, it's the order of events that matters.

From: **Mike Hearn** <mike@plan99.net>
Date: Mon, Apr 13, 2009 at 11:18 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

Oh yes, of course, that's fundamental. Silly me. Thanks for your answers. I'd recommend being over-explicit for early versions of the software, something like "Global chain is currently %d blocks long".

I guess the key problem right now is that once you generate coins, there's nobody to test it with, even for dummy transactions. Is there a plan for a mailing list or some kind of trivial marketplace to give people something to do with their newly minted bitcoins?

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Tue, Apr 14, 2009 at 7:41 PM
To: Mike Hearn <mike@plan99.net>

I started implementing a marketplace feature earlier that facilitates offering things for sale and taking orders, it's only half done though. A bit like e-bay but without auctions, just "buy now". Among other things, it would make it easy for anyone to offer currency exchange.

If you send to 1PhUXucRd8FzQved2KGK3g1eKfTHPGjgFu and e-mail me your bitcoin address, or IP if you can accept incoming connections, I'll send back the same amount +50.

From: **Mike Hearn** <mike@plan99.net>
Date: Sat, Apr 18, 2009 at 3:08 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

Hi Satoshi,

I sent you 32.51 coins, my bitcoin address is 1JuEjh9znXwqsy5RrnKqgzqY4Ldg7rnj5n

My IP is currently 84.73.233.199, however, it's a laptop so may or may not be online at the time you act on this mail. I suggest using the bitcoin address instead. It'd be convenient if the same comment functionality was available via indirect transfer. Can the comment be encrypted using the public key of the receiver and placed into a block?

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Sat, Apr 18, 2009 at 6:16 PM
To: Mike Hearn <mike@plan99.net>

I sent back 32.51 and 50.00.

I badly wanted to find some way to include a comment with indirect transfers, but there just wasn't a way to do it. Bitcoin uses EC-DSA, which was essential for making the block chain compact enough to be practical with today's technology because its signatures are an order of magnitude smaller than RSA. But EC-DSA can't encrypt messages like RSA, it can only be used to verify signatures.

From: **Mike Hearn** <mike@plan99.net>
Date: Sat, Apr 18, 2009 at 9:25 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

Thanks. I sent you back 50, so now we're even.

For some reason your transfer to me shows up as "From: unknown" even though I added you to my address book.

I have a "Generated (not accepted)" line in my transaction list, it seems like an attempt to generate a coin went wrong somehow. Not sure what happened here - presumably my node successfully solved a block but then I went offline before it was sent to the network?

I suppose for sending metadata with a transaction some other mechanism will be needed, for instance, broadcast of encrypted messages associated with a transaction that persist for (say) a month, with some kind of budget on how much storage a node can use for messages. Alternatively, a payee could generate some reference number which is of some significance to themselves but otherwise opaque, and give it to the payer, thus it does not need to be encrypted and can be put into the block directly.

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Sat, Apr 18, 2009 at 10:52 PM
To: Mike Hearn <mike@plan99.net>

Got the 50.

Transactions sent to a bitcoin address will always say "from: unknown". The transaction only tells who it's to. Sending by bitcoin address has a number of problems, but it's so nice having the fallback option to be able to send to anyone whether they're online or not. There are a number of ideas to try to improve things later. For now, if things work out like the real world where the vast majority of transactions are with merchants, they'll pretty much always make sure to set up to receive by IP. The P2P file sharing networks seem fairly successful at getting a large percentage of their users to set up their firewalls to forward a port.

The "Generated (not accepted)" normally happens if two nodes find a block at close to the same time, one of them will not be accepted. It's normal and unavoidable. I plan in v0.1.6 to hide those, since they're just confusing and annoying and there's no reason for users to have to see them. While the network is still small like it is now, if you can't receive incoming connections you're at more of a disadvantage because you can't receive block announcements as directly.

From: **Mike Hearn** <mike@plan99.net>
Date: Sat, Apr 18, 2009 at 11:23 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

Yes, I believe most P2P clients use the UPnP protocol to get routers to open up the port automatically. That would probably improve the listen rate significantly. I just discovered DMZ wasn't enabled on my router, though I thought it was. That's now fixed.

Is there a way to be told of new versions? Does the app auto update itself? Again, some kind of mailing list would be excellent.

I was thinking through how a practical micropayment implementation for the web might work in the last few days. One key issue is ensuring micropayments are fully automatic, yet can't be easily abused to drain the users account. I think the right approach would be to allow any website that presents an EV SSL cert to automatically request a micropayment, by default the browser always accepts as long as the charge is "low" and displays a small notification of what has occurred. Sites can then show that content requires payment in any way that suits their site design. Abusive sites that don't meet some simple guidelines (eg, showing unambiguously that clicking a link will trigger payment, or taking payment from direct search engine links) would simply have their SSL cert blacklisted, much like anti-phishing filters work today.

The protocol could be very straightforward and implemented by a Firefox extension or an IE BHO. Some static file (eg, a protocol buffer) is hosted on the site. It specifies the charge, a transaction description, the target IP and a URL for the browser to load after the transaction was accepted by the target node, to which the user identifier is sent in a URL parameter. The site can then give back a cookie and the paywalled content. The entire process is automatic and simply results in, say, a little coin animation in the URL bar. Thus it's as convenient as regular web browsing. The users software would have some limit on what payments are automatically accepted.

The main problem with this approach is that somebody has to decide what the user interface guidelines are, then enforce them via blacklisting, as well as decide what payment requirements are low enough to be automatic vs requiring a user prompt. This introduces a trusted authority back into the system. However, it's one that the user can choose in an open market.

By the way, if you're not already using protocol buffers for the node-to-node traffic, I recommend them. We use them here at Google for everything, they solve a lot of versioning problems simply and efficiently.

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Sun, Apr 19, 2009 at 2:14 AM
To: Mike Hearn <mike@plan99.net>

The list is:
bitcoin-list@lists.sourceforge.net
Subscribe/unsubscribe page:
http://lists.sourceforge.net/mailman/listinfo/bitcoin-list
Archives:
http://sourceforge.net/mailarchive/forum.php?forum_name=bitcoin-list

I'll always announce new versions there. Automatic update, or at least notification of new versions, is definitely on the list. There could potentially be necessary changes in the future where nobody will want to talk to you until you upgrade, and there needs to be code in the older version to convey that to the user. This is all the harder in the context of not trusting anyone.

Your approach to micropayments sounds right. At first, it might be a good idea to default to asking permission until the user gets comfortable and is ready to set it to automatic. The end goal though should get to something like you describe, where it's similar to using your cell phone without really having to think about the per minute charges.

I looked at Google protocol buffers when they were released last year, but I had already written everything by then. What I did was something similar to Boost Serialisation. For this application, where I was parsing messages from strangers who might have extreme incentive to hack the protocol, it was necessary to make it as basic as possible so I could crawl over every line of code to convince myself it was airtight. It became clear that any unnecessary degrees of freedom in the binary format multiplied the potential angles of attack. You guys are so right though to standardize across the company on protocol buffers. I think you've got the optimal solution in the general case.

From: **Mike Hearn** <mike@plan99.net>
Date: Thu, May 2, 2013 at 10:02 AM
To: satoshiarchive@gmail.com

Forwarded conversation

Subject: **Questions about BitCoin**

From: **Mike Hearn** <mike@plan99.net>
Date: Sun, Apr 12, 2009 at 12:46 PM
To: satoshin@gmx.com

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Sun, Apr 12, 2009 at 10:44 PM
To: Mike Hearn <mike@plan99.net>

Hi Mike,

I'm glad to answer any questions you have. If I get time, I ought to write a FAQ to supplement the paper.

There is only one global chain.

The existing Visa credit card network processes about 15 million Internet purchases per day worldwide. Bitcoin can already scale much larger than that with existing hardware for a fraction of the cost. It never really hits a scale ceiling. If you're interested, I can go over the ways it would cope with extreme size.

By Moore's Law, we can expect hardware speed to be 10 times faster in 5 years and 100 times faster in 10. Even if Bitcoin grows at crazy adoption rates, I think computer speeds will stay ahead of the number of transactions.

I don't anticipate that fees will be needed anytime soon, but if it becomes too burdensome to run a node, it is possible to run a node that only processes transactions that include a transaction fee. The owner of the node would decide the minimum fee they'll accept. Right now, such a node would get nothing, because nobody includes a fee, but if enough nodes did that, then users would get faster acceptance if they include a fee, or slower if they don't. The fee the market would settle on should be minimal. If a node requires a higher fee, that node would be passing up all transactions with lower fees. It could do more volume and probably make more money by processing as many paying transactions as it can. The transition is not controlled by some human in charge of the system though, just individuals reacting on their own to market forces.

Eventually, most nodes may be run by specialists with multiple GPU cards. For now, it's nice that anyone with a PC can play without worrying about what video card they have, and hopefully it'll stay that way for a while. More computers are shipping with fairly decent GPUs these days, so maybe later we'll transition to that.

A key aspect of Bitcoin is that the security of the network grows as the size of the network and the amount of value that needs to be protected grows. The down side is that it's vulnerable at the beginning when it's small, although the value that could be stolen should always be smaller than the amount of effort required to steal it. If someone has other motives to prove a point, they'll just be proving a point I already concede.

My choice for the number of coins and distribution schedule was an educated guess. It was a

difficult choice, because once the network is going it's locked in and we're stuck with it. I wanted to pick something that would make prices similar to existing currencies, but without knowing the future, that's very hard. I ended up picking something in the middle. If Bitcoin remains a small niche, it'll be worth less per unit than existing currencies. If you imagine it being used for some fraction of world commerce, then there's only going to be 21 million coins for the whole world, so it would be worth much more per unit. Values are 64-bit integers with 8 decimal places, so 1 coin is represented internally as 100000000. There's plenty of granularity if typical prices become small. For example, if 0.001 is worth 1 Euro, then it might be easier to change where the decimal point is displayed, so if you had 1 Bitcoin it's now displayed as 1000, and 0.001 is displayed as 1.

Ripple is interesting in that it's the only other system that does something with trust besides concentrate it into a central server.

Satoshi

From: **Mike Hearn** <mike@plan99.net>
Date: Mon, Apr 13, 2009 at 1:39 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

From: **Mike Hearn** <mike@plan99.net>
Date: Mon, Apr 13, 2009 at 10:51 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

Something else that isn't clear to me - does the global chain only get extended when there is actual work to do? Currently it seems to grow all the time, although there are only a few people in the network. So presumably it gets extended with null blocks. Is this actually required? The timestamping doesn't have to be actually in parallel with real time does it ... it's merely establishing an ordering of events.

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Mon, Apr 13, 2009 at 11:00 PM
To: Mike Hearn <mike@plan99.net>

Mike Hearn wrote:
My best guess - it
is the length of the global chain, and the rapid advance at the start
is as the software downloads and verifies the preceding blocks in the
chain as being valid.

Right. I'm trying to think of more clear wording for that, maybe "%d network blocks" or "%d block chain".

If we assume that 0.1% is a good risk rate, then z=5 thus
any transaction must wait a bit less than an hour before being

solidified in the chain. As micropayments for things like web content or virtual goods are by definition something that requires low overhead, waiting an hour seems like quite a significant hurdle.

For the actual risk, multiply the 0.1% by the probability that the buyer is an attacker with a huge network of computers.

For micropayments, you can safely accept the payment immediately. The size of the payment is too small for the effort to steal it. Micropayments are almost always for intellectual property, where there's no physical loss to the merchant. Anyone trying to steal a micropayment would probably not be a paying customer anyway, and if they want to steal intellectual property they can use the file sharing networks.

Currently, businesses accept a certain chargeoff rate. I believe the risk with 1 or even 0 confirming blocks will be much less than the rate of chargebacks on verified credit card transactions.

The usual scam against a merchant that doesn't wait for confirming blocks would be to send a payment to a merchant, then quickly try to propagate a double-spend to the network before the merchant's copy. What the merchant can do is broadcast his transaction and then monitor the network for any double-spend copies. The thief would not be able to broadcast during the monitoring period or else the merchant's node would receive a copy. The merchant would only have to monitor for a minute or two until most of the network nodes have his version and it's too late for the thief's version to catch up and reach many nodes. With just a minute or two delay, the chance of getting away without paying could be made much too low to scam. A thief usually needs a high probability of getting an item for free to make it worthwhile. Using a lot of CPU power to do the brute force attack discussed in the paper in addition to the above scam would not increase the thief's chances very much.

Anything that grants access to something, like something that takes a while to download, access to a website, web hosting, a subscription or service, can be cancelled a few minutes later if the transaction is rejected.

is confusing for me, because it appears the only way the honest chain can grow faster than a chain worked on by 1 attacking cpu is if the keyspace to scan looking for a partial collision is sharded evenly amongst the participating honest nodes. That way the speed at which collisions are found would be proportional to the number of nodes. Yet I don't see any discussion of such work sharding, which obviously adds complexity.

The keyspace is huge, 2^256. The thing being hashed includes the node's public key and a random nonce, so the chance of any two nodes duplicating work on the same space is negligible.

How is the required difficulty of each block communicated through the network and agreed upon?

It's not communicated. The formula is hardcoded in the program and every node does the same calculation to know what difficulty is required for the next block. If someone diverged from the formula, their block would not be accepted by the majority.

Thanks once again. I have yet more questions but this is enough for one email :) I will be happy to summarize these discussions into an FAQ-like document at some point. Apologies if the questions seem trivial.

No problem, thanks for testing it on Mac Wine.

Satoshi

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Mon, Apr 13, 2009 at 11:11 PM
To: Mike Hearn <mike@plan99.net>

It keeps getting extended all the time. If it stopped, an attacker would have time to catch up. Don't worry, empty blocks aren't very big.

As you say, it's the order of events that matters.

From: **Mike Hearn** <mike@plan99.net>
Date: Mon, Apr 13, 2009 at 11:18 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

Oh yes, of course, that's fundamental. Silly me. Thanks for your answers. I'd recommend being over-explicit for early versions of the software, something like "Global chain is currently %d blocks long".

I guess the key problem right now is that once you generate coins, there's nobody to test it with, even for dummy transactions. Is there a plan for a mailing list or some kind of trivial marketplace to give people something to do with their newly minted bitcoins?

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Tue, Apr 14, 2009 at 7:41 PM
To: Mike Hearn <mike@plan99.net>

I started implementing a marketplace feature earlier that facilitates offering things for sale and taking orders, it's only half done though. A bit like e-bay but without auctions, just "buy now". Among other things, it would make it easy for anyone to offer currency exchange.

If you send to 1PhUXucRd8FzQved2KGK3g1eKfTHPGjgFu and e-mail me your bitcoin address, or IP if you can accept incoming connections, I'll send back the same amount +50.

From: **Mike Hearn** <mike@plan99.net>

Date: Sat, Apr 18, 2009 at 3:08 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

Hi Satoshi,

I sent you 32.51 coins, my bitcoin address is 1JuEjh9znXwqsy5RrnKqgzqY4Ldg7rnj5n

My IP is currently 84.73.233.199, however, it's a laptop so may or may not be online at the time you act on this mail. I suggest using the bitcoin address instead. It'd be convenient if the same comment functionality was available via indirect transfer. Can the comment be encrypted using the public key of the receiver and placed into a block?

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Sat, Apr 18, 2009 at 6:16 PM
To: Mike Hearn <mike@plan99.net>

I sent back 32.51 and 50.00.

I badly wanted to find some way to include a comment with indirect transfers, but there just wasn't a way to do it. Bitcoin uses EC-DSA, which was essential for making the block chain compact enough to be practical with today's technology because its signatures are an order of magnitude smaller than RSA. But EC-DSA can't encrypt messages like RSA, it can only be used to verify signatures.

From: **Mike Hearn** <mike@plan99.net>
Date: Sat, Apr 18, 2009 at 9:25 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

Thanks. I sent you back 50, so now we're even.

For some reason your transfer to me shows up as "From: unknown" even though I added you to my address book.

I have a "Generated (not accepted)" line in my transaction list, it seems like an attempt to generate a coin went wrong somehow. Not sure what happened here - presumably my node successfully solved a block but then I went offline before it was sent to the network?

I suppose for sending metadata with a transaction some other mechanism will be needed, for instance, broadcast of encrypted messages associated with a transaction that persist for (say) a month, with some kind of budget on how much storage a node can use for messages. Alternatively, a payee could generate some reference number which is of some significance to themselves but otherwise opaque, and give it to the payer, thus it does not need to be encrypted and can be put into the block directly.

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Sat, Apr 18, 2009 at 10:52 PM
To: Mike Hearn <mike@plan99.net>

Got the 50.

Transactions sent to a bitcoin address will always say "from: unknown". The transaction only tells who it's to. Sending by bitcoin address has a number of problems, but it's so nice having the fallback option to be able to send to anyone whether they're online or not. There are a number of ideas to try to improve things later. For now, if things work out like the real world where the vast majority of transactions are with merchants, they'll pretty much always make sure to set up to receive by IP. The P2P file sharing networks seem fairly successful at getting a large percentage of their users to set up their firewalls to forward a port.

The "Generated (not accepted)" normally happens if two nodes find a block at close to the same time, one of them will not be accepted. It's normal and unavoidable. I plan in v0.1.6 to hide those, since they're just confusing and annoying and there's no reason for users to have to see them. While the network is still small like it is now, if you can't receive incoming connections you're at more of a disadvantage because you can't receive block announcements as directly.

From: **Mike Hearn** <mike@plan99.net>
Date: Sat, Apr 18, 2009 at 11:23 PM
To: Satoshi Nakamoto <satoshin@gmx.com>

From: **Satoshi Nakamoto** <satoshin@gmx.com>
Date: Sun, Apr 19, 2009 at 2:14 AM
To: Mike Hearn <mike@plan99.net>

The list is:
bitcoin-list@lists.sourceforge.net
Subscribe/unsubscribe page:
http://lists.sourceforge.net/mailman/listinfo/bitcoin-list
Archives:
http://sourceforge.net/mailarchive/forum.php?forum_name=bitcoin-list

I'll always announce new versions there. Automatic update, or at least notification of new versions, is definitely on the list. There could potentially be necessary changes in the future where nobody will want to talk to you until you upgrade, and there needs to be code in the older version to convey that to the user. This is all the harder in the context of not trusting anyone.

Your approach to micropayments sounds right. At first, it might be a good idea to default to asking permission until the user gets comfortable and is ready to set it to automatic. The end goal though should get to something like you describe, where it's similar to using your cell phone without really having to think about the per minute charges.

I looked at Google protocol buffers when they were released last year, but I had already written everything by then. What I did was something similar to Boost Serialisation. For this application,

where I was parsing messages from strangers who might have extreme incentive to hack the protocol, it was necessary to make it as basic as possible so I could crawl over every line of code to convince myself it was airtight. It became clear that any unnecessary degrees of freedom in the binary format multiplied the potential angles of attack. You guys are so right though to standardize across the company on protocol buffers. I think you've got the optimal solution in the general case.