Defendants
Pieter Wuille
First
13 October 2023

IN THE HIGH COURT OF JUSTICE **CLAIM NO. IL-2022-000069**
BUSINESS AND PROPERTY COURTS OF ENGLAND AND WALES
INTELLECTUAL PROPERTY LIST (CHD)

BETWEEN

DR CRAIG STEVEN WRIGHT & Ors

Claimants

—and—

BTC CORE & Ors

Defendants

WITNESS STATEMENT OF
PIETER WUILLE

I, **PIETER WUILLE**, of c/o Macfarlanes LLP, 20 Cursitor Street, London EC4A 1LT, WILL SAY as follows:

1 I am the Fourth Defendant in these proceedings. I am making this witness statement in relation to the trial of a preliminary issue in the case, namely whether the First Claimant, Dr Wright, is the pseudonymous inventor of Bitcoin, Satoshi Nakamoto.

2 This witness statement has been prepared following two videoconference calls with my solicitors, Macfarlanes LLP (over which privilege is not waived). After each call, Macfarlanes prepared written drafts of this statement, which I reviewed and amended before signing the Statement of Truth below.

3 Unless otherwise stated, the facts and matters set out in this witness statement are within my own personal knowledge and recollection and I believe them to be true. Where the facts and matters are not within my knowledge, I have given the source of my belief and I believe them to be true.

4 Some of the events set out in this statement took place a significant time ago. I have set out those events to the best of my recollection. Where I cannot recall details precisely, I have said so. In order to make this statement, I have refreshed my memory by:

4.1 reviewing certain disclosed documents pointed out to me by Macfarlanes;

4.2 reviewing certain of the witness statements filed in the case, again pointed out to me by Macfarlanes; and

4.3 reviewing other documents, such as blog posts which I recall reading at the times of events that Macfarlanes have asked me about, or the log of changes ("commits") made to the Bitcoin software code, which records when and by whom certain changes were made to the code.

5 I have included these documents in the list appended to the end of this statement. I have indicated below the distinction between matters I remember independently, and those where my memory has been refreshed.

My background

6 I have always been interested in computers. I learned to code in BASIC when I was about 11 and started learning C when I was 16 or 17. I graduated with "great distinction" from KU Leuven University with a masters degree in Engineering, specialization Computer Science, in 2007, and in 2011 obtained a PhD in the same field, awarded by both KU Leuven and Ghent University. My thesis was titled "Functional Programming Abstractions for CP Modeling", and can be found online. I learned to code in Java at university and, from my involvement in the Bitcoin project, learned to code in C++.

My involvement with Bitcoin

7 I discovered Bitcoin on the Freenode IRC network, in a channel called '#haskell-blah' (related to the Haskell programming language, which I used in my PhD) in around December 2010. I started contributing to the project in my free time in early 2011. I got involved because Hal Finney had posted a puzzle on the Bitcoin.org/forum where you could win 20 BTC (increased to 50 BTC the next day)[1]. It was posted on 19 February 2011 and was won a day later by Mike Hearn.

8 I tried to solve the puzzle by creating a patch for the reference Bitcoin software, to allow it to accept externally generated keys. That patch became my first contribution to the Bitcoin software. I worked up the patch and opened a "pull request" (i.e. a request for a change to the software to be included in the source code). Having reviewed that pull request in preparing this statement, I see that it was on 12 March 2011.

9 When writing the patch, I discussed it with Gavin Andresen (who was at that time the lead developer of the Bitcoin open source software) through the bitcoin.org forum between around 8-10 March 2011. I asked Gavin about a roadblock I hit in development and Gavin said that

[1] See https://bitcointalk.org/index.php?topic=3638.0. I reviewed this thread to find the specific date of the puzzle, as well as the amounts involved.

he forwarded my question to Satoshi. On 10 March 2011, Gavin forwarded me Satoshi's response by email, which included a suggested patch (which he described as untested and, as far as I recall, didn't work without changes). The next day Gavin forwarded another response from Satoshi with some further comments. I reviewed the forum messages which are still accessible (I believe there were others at the time) and those emails to prepare this statement and copies are exhibited at PW1/1-5.

10 In late April 2011, Gavin Andresen asked me to join the maintainer team for the Bitcoin software (at the time this was called the core developer team). The maintainers have the ability to accept proposed changes to the software and include them in the source code. I think that Gavin was looking for someone to help, and saw I was actively contributing and learning my way around the software, so thought I would be a valuable addition: when he made me a maintainer he emailed me saying: "*Thanks for all your work bugfixing and creating new features-- I'm going to assume that you'd be willing to help pull patches, so I made you part of the team*." I reviewed that email to prepare this statement, and a copy is exhibited at PW1/6. The first time I used my ability to accept a patch was on 1 May 2011. I searched through the history of pull requests to remember the specific date I first used my ability to accept a patch (though did recall it was in May 2011).

11 At that time, the project was smaller and there was less of a distinction between contributors and maintainers. As the project grew and developed, the standards around maintaining the project evolved too. Today, the role of a maintainer mainly involves judging whether a contribution has had enough review from the right people. How much review, and who by, will depend on the patch. A very invasive patch may not be accepted until there is widespread agreement, but a simple one can be accepted quickly. I left the maintainer team on 8 July 2022[2], but I continue to be an active contributor to the project. Initially I just worked on the software in my free time, but since September 2014 contributing to Bitcoin has been part of my job (first for Blockstream, since 2020 for Chaincode Labs).

12 In the rest of this statement I describe certain concepts related to Bitcoin and the Bitcoin software, and terminology commonly used during my years of involvement as a voluntary contributor to the software. I am aware of these concepts and this terminology from my work on the Bitcoin software and my involvement in the development of Bitcoin.

"Bootstrapping"

13 "Bootstrapping" is the process by which a node first connects to the peer to peer network of nodes. The term "peer to peer architecture" was used very early on by Satoshi to refer to the

[2] I searched through the history of pull requests to refresh my memory of the specific date I stopped being a maintainer.

design of Bitcoin. It refers to a network architecture where all participants perform similar functions, which contrasts heavily with a traditional server-client architecture.

14 Bootstrapping is essentially the process of getting a new node its first foothold into the network. The aim is to allow a new node to connect to at least one other honest node.

15 Over time, Bitcoin has employed a number of bootstrapping mechanisms. The first was IRC seeding, which as far as I know was included in the original software[3]. IRC is an internet chat protocol, and the mechanism worked by automatically connecting to an IRC channel on a particular IRC server (which were hardcoded into the software) and seeing which other nodes are in the channel, while also watching for nodes that join later. It then built a database of IP addresses of nodes.

16 In preparing this statement, I have reviewed the pull requests to see how the bootstrapping mechanism developed. I have seen that, in June 2010, in version 0.2.12 of the software, hardcoded seed IP addresses were added. The developers made a list of IP addresses which seemed to be Bitcoin nodes, and these were hardcoded into the software as a backup to which the software could connect, in case the IRC mechanism failed for any reason.

17 DNS seeding then replaced IRC seeding over time. In June 2011, Jeff Garzik (who was a core developer at the time) introduced DNS seeding. This mechanism connects the node to a DNS server that records a number of good Bitcoin node IP addresses that the software can connect to. Initially DNS seeding ran alongside IRC seeding. In June 2011, in version 0.3.22, the IRC seeding code was revised to connect to one of 100 IRC channels (chosen at random) to improve performance as the Bitcoin network had expanded. In version 0.3.24 DNS seeding was enabled by default and in version 0.6.0, IRC seeding was disabled by default (although it remained present in the code). Two years later in version 0.9.0 in March 2014, support for the IRC mechanism was removed from the code base entirely. Whilst I know the above is correct having looked at the pull requests, the history also reflects my general recollection of the development of these mechanisms.

18 Bootstrapping is the mechanism for a new node to make a first connection into the network. Once the connection is made, the Bitcoin software also includes a mechanism for a node to request more known IP addresses of nodes on the network using the "gossip" functions: "getaddr" and "addr". By sending a getaddr message, a node can request another node provide a list of known IP addresses. The recipient node responds with one or more "addr" messages, each containing one or more IP addresses. This allows nodes to expand their connections in the network.

[3] I reviewed the commit history to verify that IRC seeding was indeed part of the original software release.

19 The way that a node responds to a getaddr request has changed over time but, at high level, the recipient node will send the list of IP addresses of peers it knows about (initially all of them, but as explained below, later only a subset). I have reviewed the pull requests to refresh my memory of the details of the development of these functions. In June 2010, in version 0.2.10 of the software, in a change made by Satoshi, the code was updated so that no more than 1,000 IP addresses would be processed at once: i.e., when responding to a getaddr request, if the recipient node knew more than 1,000 IP addresses, it would respond with multiple addr messages each containing up to 1,000 IP addresses. The requesting node would also ignore messages that contained more than 1,000 addresses. That was changed again in October 2010 in version 0.3.14 of the software. From that point on, if the recipient node knew a sufficient number of addresses only a random subset would be sent to the requesting node. It was randomised but, on average, 2,500 IP addresses would be shared (again, split into addr messages each containing no more than 1,000 IP addresses).

20 The information shared through the gossip functions includes “service flags”, which announce what services each node provides. In the original Bitcoin software, there was only one service (“NODE_NETWORK”), and so all nodes set the “NODE_NETWORK” flag. As far as I remember, it was only with the release of BitcoinJ in 2011 that software appeared which did not offer the “NODE_NETWORK” service. The original Bitcoin software source code included some unfinished elements that appeared to envision a light-weight “client mode” that would not set the “NODE_NETWORK” flag. Besides this, there was no way for the gossip and bootstrapping mechanism to distinguish between types of nodes (e.g., nodes being run commercially, or nodes with a large number of connections, or mining/non-mining nodes). A few more services have been added to the protocol since around 2014, but it remains the case that no distinction between the types of nodes listed above exists. Generally all nodes can offer any services they want, and none have a special position, making them all peers. This is what sets it apart from more traditional client-server architectures.

“Whitelisting”

21 The concept of “whitelisting” nodes was something I introduced into the Bitcoin software in pull request 4378, which was opened on 21 June 2014. I have reviewed that pull request to refresh my recollection as to the time it was introduced. Looking at the pull request history, it appears that there was an earlier attempt in December 2013 that also tried to introduce the term "whitelisting" of nodes (pull request 3403), though that pull request was abandoned in favour of my later one. The term did not occur anywhere in the code base before that time, as far as I know. Whitelisting means that certain nodes can be made exempt from some of the protections and connection limits applied by the software. These, e.g., limit the size of messages and the number of transactions a node can announce. These limitations were originally in place for a number or reasons, but generally to limit functions that risked being

used for denial of service attacks, or for privacy reasons. Many of those connection limits have been in the software for as long as I can remember.

22 Whitelisting was directed at facilitating a situation where one person controls multiple nodes, or multiple nodes are operated by people who have a business relationship with each other and so can be trusted not to attack each other, by bypassing certain of those limits. For example, you might have one node connected to a public network, and a number internally that trust each other, so you can whitelist those internal nodes and the transaction limits would not apply between them. Whitelisting would also override "misbehavior limits". In previous versions of the software, nodes were given a "misbehavior" score, and if they hit a certain score, they were banned, meaning it would not be allowed to connect to the node that banned it. However, if a node operator whitelisted a certain node, it would not be barred from connecting, irrespective of its misbehavior score. The concept of a "misbehavior" score was introduced by Gavin Andresen in 2011 (in pull request 517, which I reviewed for the purpose of this statement). It is still the case that nodes are given such a misbehavior score, but when the threshold score is reached the node is no longer disconnected and banned; instead its connection becomes eligible for replacement by new connections from non-misbehaving nodes.

23 As far as I know, there was no analogous concept to this before I introduced it, and the term "whitelisting" was not used for anything else in Bitcoin. In addition, the bootstrapping mechanisms I have described above have nothing to do with trusting or whitelisting nodes. They are very different concepts. Nodes are not universally incentivised to maximise their number of connections or to connect to certain other types of nodes.

CheckBlockHeader

24 The "CheckBlockHeader" function checks whether a received block header is valid. In particular, it does context free checks of BlockHeader validity, meaning it checks the things that can be checked before knowing the details of the parent blocks to the block being checked. These include, for example, checking whether the block header meets the proof of work requirement.

25 I introduced the CheckBlockHeader function as part of the header synchronisation changes I introduced in version 0.10 of the Bitcoin software on 11 March 2014 in a commit titled "Split up CheckBlock in a block and header version". I have reviewed that commit to refresh my memory about how and when this function was introduced. As the commit name suggests, it split the functionality that was previously present in the CheckBlock function over a (now trimmed down) function CheckBlock and a new function CheckBlockHeader. I believe the term CheckBlockHeader was not used before that time.

OP_ operations

26 The Bitcoin software supports scripts written in Bitcoin script. Scripts are made up of operations specified by Operation Codes, or "opcodes". These allow users to specify different operations with Bitcoin and Bitcoin transactions. Bitcoin Script has similarities with the Forth programming language, although it also includes some additional opcodes linked to transaction validation.

27 "OP_MUL" and "OP_DIV" are opcodes that, in previous versions of the Bitcoin software, allowed scripts to perform multiplication and division operations, respectively. Those functions were for integer mathematics, not modular arithmetic. I do not believe the Bitcoin software ever supported modular arithmetic, beyond including a function to calculate a modulus ("OP_MOD"). OP_MUL, OP_DIV and a number of other operations were disabled in August 2010 – I believe because there were concerns that they could possibly be used for denial of service attacks (along with OP_CAT, which was a known denial of service attack risk) and because they were not being used. The disabled opcodes were never replaced. I have reviewed the commit removing these functions to refresh my memory about how and when they were removed.

28 Other opcodes include "OP_CLTV". This was introduced in around 2016. It allows a user to create a Bitcoin script which makes a transaction valid only once a certain time or certain block height is reached. It works by checking the lock time associated with the transaction. All transactions have a lock time value that can be set by a user, but until OP_CLTV was introduced, there was no way to access that value and check it in a script. In practice, this allows users to, e.g., create a UTXO which can be spent after a specific time.

"**UTXOs" and the "unconfirmed transaction pool"**

29 A "UTXO" is an "unspent transaction output". This is a specific concept in the Bitcoin protocol. So, "a UTXO" is an output which has not been spent yet, while "the UTXO model" is a term now used to describe a ledger design like Bitcoin's that tracks specific coins, which must be spent in their entirety (as opposed to an "account model" that involves balances which can increment and decrement).

30 Originally, to validate transactions, the software used a transaction index database, with one entry per transaction ever created, containing information about where it could be found on disk, and which of its outputs were spent and when. However, in version 0.8 of the software, we made a major update and changed the software to explicitly use a UTXO model. That meant replacing the transaction index with a database containing just the unspent transaction outputs (i.e., the UTXOs), against which new transactions were (and are) validated. That is because a transaction can only be validated against a UTXO; a spent transaction output cannot be spent again so there is no need to check against spent outputs. This change

resulted in a major performance improvement, because: (a) the database was much smaller now that it no longer contained information about spent outputs, and (b) it no longer needed to look up the full list of transactions in the blockchain on disk (all information necessary for validation was in the UTXO database).

31 I was the author of the patch that made this change, which I introduced by pull request 1677. I am not aware that the term UTXO was used much before version 0.8 of the software was introduced – the concept was only first becoming more important around then. I believe that Alan Reiner, who is the author of the now-defunct BTC wallet software "Armory", was the first person to use it. I remember from discussions at around this time with Alan and Andrew Miller (who is now an Assistant Professor at the University of Illinois, Urbana-Champaign) that Alan was thinking about creating a Merkle tree over all UTXOs to prove coins had not been spent, and so as far as I'm aware he was the first person who needed to refer to an unspent transaction output often enough to need a shorthand. On 21 June 2012, Alan (whose nickname was 'etotheipi') commented in the #bitcoin-dev channel "*btw, I'm going to start using utxo to refer to unspent-txout*"[4] and that was the first time I saw the term. I reviewed that message in order to refresh my memory about when and how this came about. Having reviewed my IRC logs for the purpose of preparing this statement, I have seen that, in an unconnected discussion some months later, Jeff Garzik asked me what UTXO meant – which shows that it was not a well-established term.

32 UTXO caching was introduced in the same change in 0.8, and has been extended upon and improved many times since. UTXO caching is the process by which the software keeps a subset of the UTXO database cached in memory for faster access. It was designed by me. It is an unusual design which exploits the property that the UTXO is only spent once and there are no writes to UTXOs.

The Mempool

33 A node's "mempool" is the set of unconfirmed transactions which have been relayed to a miner but have not made it into blocks (yet). Miners fetch transactions from their mempool to record in blocks. I believe that the specific abbreviation "mempool" (for "memory pool") was introduced by Jeff Garzik in the codebase in 2012 in pull request 1095; for the purpose of this statement I conducted a search through the bitcointalk forum and IRC logs, and the abbreviation does not appear to have been used before 2012.

34 I believe that in original software, the mempool was unlimited, so transactions would simply stay in the mempool forever if they were not included in blocks. That became a problem because nodes would run out of memory to hold the mempool transactions, and would crash.

[4] The old logs from #bitcoin-dev are available here: https://buildingbitcoin.org/bitcoin-dev/log-2012-06-21.html.

The software was updated with two methods to deal with this problem: eviction and expiration. Expiration is the process by which a transaction that has not been confirmed for a specified period drops out of the mempool automatically. This allows the owner of the BTC being spent to reuse their coins. The code for the expiration process was introduced in pull request 6722 in 2015 by Matt Corallo (in fact I left a positive review on the pull request). I have reviewed that pull request to refresh my memory about how and when the software was updated.

The Bitcoin alert system

35 The early versions of the Bitcoin software included an alert system. It essentially allowed a pop-up message to be sent to all Bitcoin clients, and it could disable a few RPCs (remote procedure calls) related to sending coins, which created a sort of safe mode. Its purpose was as a warning mechanism. For example, it could be used to alert users to some critical bug where we needed to quickly alert the whole ecosystem to a problem.

36 Satoshi wrote on the bitcoin.org forum when the alert system was introduced that it was temporary.[5] I reviewed the commit history to prepare this statement and the ability for the alert system to disable RPC commands was removed in December 2010, in version 0.3.19, by Satoshi, in commit 986b5e257e2bb9d7aaed5111ca335732f8808b2d.

37 The alert system was used a number of times between 2012 and 2014. I recall it being used during March 2013. Up to version 0.7 of the Bitcoin software, a latent restriction was present that would cause it to unintentionally reject large blocks with certain properties. That bug was unknowingly removed in version 0.8 of the software, causing a fork (separate blockchains) between users of version 0.8 of the software, and users of earlier versions. After identifying what happened, the alert system was used to notify people (specifically asking them to temporarily revert to version 0.7). This issue was described in BIP50. I have reviewed a webpage on the alert system to refresh my memory of the times the alert key was used.

38 The alert was a very weak protection at best, and I recall that Satoshi referred to it as unimportant. In a forum post (which I reviewed to refresh my memory), he said: "*WRT the alert system, who cares? The most the key can do is temporarily disable six json-rpc commands until the site owners either add the -disablesafemode switch or upgrade. All nodes keep running and generating, the network stays up. If I'm not available, any script kiddie can figure out how to add two characters and make a new version that disables the alert system. It would be a temporary inconvenience only*" and "*It can't shut down the complete network*."[6] As far as I know, the alert system never allowed transactions to be blocked or funds to be frozen.

[5] https://bitcointalk.org/index.php?topic=2228.msg29479#msg29479
[6] https://bitcointalk.org/index.php?topic=898.msg11155#msg11155

39 There was one specific key that you needed to send the alert message and every alert needed to be signed by that key. Satoshi gave this to a few people. One of the reasons for retiring the alert system was that too many people got access to that key, and it was unknown who had access to that key, and because the developers thought it was inappropriate to have a warning system for one client hard-wired into the protocol.

40 In pull request 7692, the code supporting the alert system was completely removed. I reviewed that pull request to refresh my memory of when and how the system was updated.[7]

Block sizes

41 Currently, a block in the Bitcoin blockchain is typically 2MB, although the consensus rules allow it to be up to 4MB. Before the 2017 "SegWit" software update, of which I was a co-author, and since the 0.3.12 release in September 2010 by Satoshi, the limit was 1MB. The SegWit update covered a number of changes to the Bitcoin software code – most importantly fixing a transaction malleability issue. However, we also took the opportunity to introduce a block size increase.

42 Essentially, the block size limit is a trade off between the ability for users to transact (because the difficulty of the proof of work required to find a block is calibrated to find a block about every 10 minutes) and the ability for users to verify that everyone is acting honestly (because a greater number of users have the computing power to validate smaller blocks). Every transaction in a block needs to be validated, so if you make blocks extremely small: everyone can validate everything on the simplest hardware but only very few people can make transactions. If you make blocks extremely large it would be very easy to anyone to transact, but very hard for anyone to validate the system is honest. Capacity limit is a trade off between those. Regarding the extreme of allowing many people to transact, but few to validate, many systems like this already exist. In my opinion, Bitcoin is interesting because it allows pursuing a different trade-off. However, history has shown that opinions on this matter differ wildly.

43 There has been discussion of increasing the block size further in Bitcoin, but as far as I am aware no serious steps have been taken to make a change – I believe because of the trade offs identified above. I would find the suggestion of changing the Bitcoin protocol to allow very large blocks, e.g. 1TB, outlandish. I cannot see that a block of that size could be practically verified across the network: most people do not have the type of internet that would allow them to download a terabyte of data every 10 minutes. It is also unfeasible for most users in terms of computation and storage costs.

Block explorer

[7] I reviewed the following to refresh my memory on the alert system retirement timeline: https://bitcoin.org/en/alert/2016-11-01-alert-retirement

44 A "block explorer" is a website that allows users to inspect transactions that have happened on the Bitcoin blockchain. The first block explorer was called "blockexplorer.com", and was the origin of the term (as far as I am aware). The site is still in operation, although it now looks nothing like the original version. However, what appears (as far as I recall) to be the original version of the website is available on the internet archive (which shows the earliest version of blockexplorer.com on 21 November 2010). Since then, a number of alternative tools have been created, which are also referred to "block explorers".

ECDH, DSA and ECDSA

45 "ECDH" stands for "Elliptic-curve Diffie–Hellman". ECDH is a key agreement protocol or exchange protocol – i.e., a way for two parties who both have a public and corresponding private key to create a shared secret that only they can compute – even if third parties have seen their communications. It allows for encrypted communication. Bitcoin does not currently use it, although a recent protocol improvement proposal which I co-authored, BIP324, is intended to introduce ECDH, and encrypted communication, to the Bitcoin P2P protocol. Use of ECDH will be optional and negotiated on a per-connection basis between nodes to protect their individual communication. It is not, and has never been, part of the block or transaction validity rules.

46 DSA is a specific algorithm, a variant of the Schnorr signature algorithm, created to work around Claus-Peter Schnorr's patent. It is an alternative to RSA, which is a much better known algorithm that can be used for both encryption and digital signatures. DSA can only be used for digital signatures.

47 ECDSA is an elliptic curve variant of DSA. It takes DSA and substitutes the mathematical group with an elliptic curve group. As a result you get an algorithm that has smaller signatures and higher security with the same computation cost. DSA has larger keys than ECDSA. ECDSA over the secp256k1 curve was the only signature algorithm supported in Bitcoin until the adoption of BIP340 in November 2021. BIP340 (which I co-authored) added support for (a variant of) Schnorr signatures over the same curve.

48 ECDH and ECDSA are not comparable. You can't substitute one for the other. ECDH is a means for constructing shared secrets between two parties to use as an encryption key. ECDSA is a way to say a certain party has agreed to a particular statement.

Unit names

49 The term "BTC" is common shorthand for the bitcoin currency (I have used it in that sense in this statement), and sometimes for the Bitcoin system. "BCH" refers to Bitcoin Cash. Bitcoin Cash was created when the Bitcoin Cash ABC software was released, I believe in around 2017. That software created the most noteworthy fork in the Bitcoin blockchain. I cannot think

of any other use of these terms together by the Bitcoin community (although BCH is also shorthand for the Bose-Chaudhuri-Hocquenghem error correcting code which was incorporated into the new Bitcoin address format introduced in 2017).

50 Bitcoin Core is the current name of the most commonly used (but not only) fully-validating node software implementation. I recall that the name Bitcoin Core Client was introduced in version 0.9 of the software, in 2014. Before that the software was just called the "Bitcoin Client". The new name was introduced specifically because we wanted a distinction between the software project and the protocol itself. I remember the name being suggested by Gavin Andresen. I do not recall it was ever used before that software update.

51 "A Satoshi" is a common term used to refer to the smallest unit into which a Bitcoin can be divided – i.e., the smallest units recognised by the Bitcoin protocol. It is equivalent to one 100-millionth of a Bitcoin. I do not recall the term "a Satoshi" being used at all in the early days of Bitcoin. At that time, Bitcoin was much lower value in the early years, and so there was no need to talk about such small units. In fact, until 2011, I think that the Bitcoin software only displayed two decimal digits. I reviewed the commit history for the purposes of preparing this statement and, based on that review, I believe the change to show the full 8 decimals was made by Gavin Andresen, in February 2011, in version 0.3.21.

Satoshi Nakamoto and Dr Wright

52 I was aware of Satoshi Nakamoto from my earliest work on the Bitcoin software, although by that time he had stopped making public comments on the project. As I discuss above at paragraph 7, I received a couple of emails from Gavin Andresen forwarding emails which he said had been sent to him by Satoshi.

53 I think the first time I became aware of Dr Wright was around the time he posted a screenshot in a blog that was supposed to be a message signed with one of Satoshi's keys. I remember reading this blog post when it first came out, and reading articles responding to it which argued that it was not a genuine signature, and instead reused an existing, public signature by Satoshi from the bitcoin blockchain. I remember that I looked at the blog post and myself verified that it took an existing signature by Satoshi and converted it into OpenSSL format rather than the Bitcoin format so it didn't look the same as the original. The most obvious tell is that the signature could not be identical to one that was already used. In short, the signature in the blog post proves nothing; I formed the view that it was a deliberate attempt at making an old signature look like it was a recent one.

54 Having been aware of Satoshi's contribution, disappearance and apparent intentional step back from the Bitcoin project, I would have been sceptical of anyone claiming publicly to be him (or them) in the way that Dr Wright did. But on top of that, I remember that when I reviewed

the blog, it convinced me Craig Wright was not Satoshi, and since that point in time I think I would have needed very convincing evidence to change my mind about this.

Confirmation of compliance

I understand that the purpose of this witness statement is to set out matters of fact of which I have personal knowledge.

I understand that it is not my function to argue the case, either generally or on particular points, or to take the court through the documents in the case.

This witness statement sets out only my personal knowledge and recollection, in my own words.

On points that I understand to be important in the case, I have stated honestly (a) how well I recall matters and (b) whether my memory has been refreshed by considering documents, if so how and when.

I have not been asked or encouraged by anyone to include in this statement anything that is not my own account, to the best of my ability and recollection, of events I witnessed or matters of which I have personal knowledge.

Statement of Truth

I believe that the facts stated in this witness statement are true. I understand that proceedings for contempt of court may be brought against anyone who makes, or causes to be made, a false statement in a document verified by a statement of truth without an honest belief in its truth.

Signed: DocuSigned by: Pieter Wuille F17A5EE97D82485... **PIETER WUILLE**

Dated: 13 October 2023

Certificate of compliance

I hereby certify that:

1. I am the relevant legal representative within the meaning of Practice Direction 57AC.

2. I am satisfied that the purpose and proper content of trial witness statements, and proper practice in relation to their preparation, including the witness confirmation required by paragraph 4.1 of Practice Direction 57AC, have been discussed with and explained to Pieter Wuille.

3. I believe this trial witness statement complies with Practice Direction 57AC and paragraphs 18.1 and 18.2 of Practice Direction 32, and that it has been prepared in accordance with the Statement of Best Practice contained in the Appendix to Practice Direction 57AC.

.....

Name: C. CHARLTON

Position: Partner, Macfarlanes LLP

Date: 13 Oct, 2023

APPENDIX

List of documents the witness has referred to or been referred to for the purposes of providing this statement.

Document ID:

1. ID_000504
2. ID_003994
3. ID_000549
4. ID_000550
5. ID_000568
6. ID_003565
7. ID_000551
8. ID_003840
9. ID_003998
10. ID_004009

Witness Statements:

11. The first witness statement of David Bridges
12. The first witness statement of Robert Jenkins
13. The first witness statement of Ignatius Pang
14. The first witness statement of Maxwell Lynham
15. The first witness statement of Stefan Matthews
16. The first witness statement of Dr Craig Steven Wright

Bitcoin Improvement Proposals:

17. BIP324 (accessible at https://github.com/bitcoin/bips/blob/master/bip-0324.mediawiki)
18. BIP340 (accessible at https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki)

Commits:

19. Commit 4bd188c4383d6e614e18f79dc337fbabe8464c82 (accessible at https://github.com/bitcoin/bitcoin/commit/4bd188c4383d6e614e18f79dc337fbabe8464c82)
20. Commit 986b5e257e2bb9d7aaed5111ca335732f8808b2d (accessible at https://github.com/bitcoin/bitcoin/commit/986b5e257e2bb9d7aaed5111ca335732f8808b2d)

Chat logs, messages and emails:

21. Fwd: fSpent and importing wallets… [**PW1/2-5**]

22. Public archive of chat logs from #bitcoin-dev, 21 June 2012 (accessible at https://buildingbitcoin.org/bitcoin-dev/log-2012-06-21.html)

23. Public archive of chat logs from #bitcoin-dev, 20 August 2012 (accessible at https://buildingbitcoin.org/bitcoin-dev/log-2012-08-20.html)

24. Re: Import backed up wallet. [**PW1/1**]

25. Re: You willing to help with PULL requests? [**PW1/6**]

Pull Requests:

26. PR116 (accessible at https://github.com/bitcoin/bitcoin/pull/116)

27. PR178 (accessible at https://github.com/bitcoin/bitcoin/pull/178)

28. PR517 (accessible at https://github.com/bitcoin/bitcoin/pull/517)

29. PR1095 (accessible at https://github.com/bitcoin/bitcoin/pull/1095)

30. PR1677 (accessible at https://github.com/bitcoin/bitcoin/pull/1677)

31. PR3203 (accessible at https://github.com/bitcoin/bitcoin/pull/3203)

32. PR3403 (accessible at https://github.com/bitcoin/bitcoin/pull/3403)

33. PR4378 (accessible at https://github.com/bitcoin/bitcoin/pull/4378)

34. PR6722 (accessible at https://github.com/bitcoin/bitcoin/pull/6722)

35. PR7692 (accessible at https://github.com/bitcoin/bitcoin/pull/7692)

Web pages (including blog posts online forums):

36. Added some DoS limits, removed safe mode (0.3.19) (accessible at https://bitcointalk.org/index.php?topic=2228.msg29479#msg29479)

37. Alert system (accessible at https://en.bitcoin.it/wiki/Alert_system)

38. Alert System Retirement (accessible at https://bitcoin.org/en/alert/2016-11-01-alert-retirement)

39. A complete history of Bitcoin's consensus forks (accessible at https://blog.bitmex.com/bitcoins-consensus-forks/)

40. Development of alert system (Satoshi's response is accessible at https://bitcointalk.org/index.php?topic=898.msg11155#msg11155)

41. How did peer discovery work in Bitcoin v0.1? (my response is accessible at https://bitcoin.stackexchange.com/questions/119507/how-did-peer-discovery-work-in-bitcoin-v0-1/119509#119509)

42. Op Ed: How Many Wrongs Make a Wright? (accessible at https://bitcoinmagazine.com/business/op-ed-how-many-wrongs-make-wright)

43. Prize for importing private key (accessible at https://bitcointalk.org/index.php?topic=3638.0)