IN THE HIGH COURT OF JUSTICE
BUSINESS AND PROPERTY COURTS OF ENGLAND & WALES
INTELLECTUAL PROPERTY LIST (ChD)

Claim No: IL-2021-000019

B E T W E E N :

CRYPTO OPEN PATENT ALLIANCE — **Claimant**

-and-

DR CRAIG STEVEN WRIGHT — **Defendant**

EXPERT REPORT OF

PROFESSOR SARAH MEIKLEJOHN

Table of Contents

Background

1. Before diving into the technical aspects of this report, I'll set out my own background and expertise in the subject matter. In terms of my academic credentials, I have a Bachelor of Science in Mathematics and a Master of Science in Computer Science from Brown University, and a PhD in Computer Science from the University of California, San Diego. After finishing my PhD in April 2014, I started as a Lecturer at University College London in September 2014, where I was promoted to Reader in October 2017 and Professor in Cryptography and Security in October 2020. My CV and list of publications are annexed to this report at Annexes SM1 and SM2 respectively.

{F/163}
{F/164}

2. One of my primary academic fields and areas of expertise is cryptography. I have developed various types of cryptographic primitives: zero-knowledge proofs, digital signatures, verifiable secret sharing schemes, anonymous credentials, and more. To date, I have published 51 peer-reviewed articles in international journals, conferences, and workshops.

3. From the very start of my academic career, I have been interested in electronic cash. This is perhaps because I did my Master's degree with Anna Lysyanskaya, who co-authored some of the seminal papers on this topic. For example, my first published paper was entitled "ZKPDL: A Language-Based System for Efficient Zero-Knowledge Proofs and Electronic Cash" (**Exhibit SM-1)** and described a library for electronic cash that my collaborators and I implemented in C++.

{H/173}

4. I first learned about Bitcoin during my PhD. The first time I heard about it was in passing in the summer of 2011, but it wasn't until April 2012 that I began to explore it in depth. The first academic publication I wrote looking at Bitcoin was an empirical analysis of the extent to which its users achieved anonymity. This research, which I worked on in a nearly full-time capacity for over a year, led me to interact closely with data from the Bitcoin blockchain; to manually perform hundreds of my own transactions with tens or hundreds of different counterparties; to buy bitcoins from various sources; to read about Bitcoin to understand it at a conceptual level; and even to participate in the Bitcoin mining process. As such I gained an extensive and hands-on understanding of how Bitcoin worked.

5. The resulting paper was published in 2013 and is today one of the most highly cited academic papers on Bitcoin (with over 1700 citations to date). Furthermore, the research attracted attention in the media and elsewhere, which led to me doing follow-on hands-on explorations

of specific behaviours and events (e.g., investigating thefts or specific Bitcoin-based services).

6. After working on this research, I became quite interested in Bitcoin, cryptocurrencies, and blockchains in general. To date, I have published 22 peer-reviewed articles on the direct topic of blockchains, and many more that have applications to this space (e.g., papers that develop new cryptographic primitives whose properties are well suited to usage in a blockchain). I also taught a 10-week Cryptocurrencies module for our Masters in Information Security course in 2019. I rarely interact with the Bitcoin software directly anymore, but my students do, and I have kept up with its conceptual changes over the years. I do, however, engage directly with open-source software development in general.

My role as an expert witness and my instructions

Duties and Independence

7. I have been instructed by Bird & Bird, on behalf of the Crypto Open Patent Alliance ("COPA"), to undertake the role of expert witness in these proceedings. Bird & Bird have brought my attention to Part 35 of the Civil Procedure Rules 1998, the Practice Direction which supplements Part 35 and a document issued by the Civil Justice Council titled "Guidance for the instruction of experts in civil claims." Bird & Bird has also provided me with an excerpt from a case called "The Ikarian Reefer" headed "The duties and responsibilities of expert witnesses." I confirm that I have read these documents and understand my duty to assist the Court. I understand that this duty overrides any obligation to COPA or Bird & Bird and I have approached my analysis from this perspective, being impartial. I confirm that I have complied and will continue to comply with that duty. I also confirm that the opinions expressed in this report are my own.

8. Bird & Bird have informed me that the parties are engaged in proceedings relating to the identity of the creator of Bitcoin and author of the Bitcoin whitepaper, and whether or not Dr Craig Wright is the pseudonymous creator, Satoshi Nakamoto, of Bitcoin and author of the paper. Having worked in the area of Bitcoin and cryptocurrency for over a decade, I am familiar with the background to its creation (and was broadly aware of Dr Wright's claims to be Satoshi Nakamoto), but do not offer any insight into the answer to that question or have an opinion on whether or not Dr Wright is Satoshi Nakamoto, other than in the context of the

documents with which I have been provided and the questions on which I have been asked to opine. I was not aware of the current proceedings prior to being contacted by Bird & Bird.

9. I confirm that the fees I am receiving are not dependent in any way on the outcome of these proceedings, and that the view expressed in this report have not been influenced by those fees in any way.

10. Prior to my role as an expert in these proceedings, I have not acted as an expert witness in the UK (or any other jurisdiction).

Scope of my report

11. Before being asked to prepare my report, Bird & Bird asked me to prepare a "technical primer" setting out the basic background on blockchain and cryptocurrency technology (including Bitcoin), including the following topics:
    a. The origins of Bitcoin
    b. How the Bitcoin blockchain works in general terms
    c. What each block consists of
    d. The coding of the genesis block
    e. Digital signatures
    f. Public/private keys
    g. How keys are held/retrieved
    h. Wallets
    i. Shamir secret sharing (as a means of holding private keys)
    j. Anatomy of a transaction
    k. The choice of Bitcoin's elliptic curve
    l. Forks

12. I was instructed to provide feedback on two versions of the "technical primer" prepared on behalf of Dr Wright, dated 3 May 2023 and 27 June 2023. I was subsequently instructed that the technical primer was not agreed, and that as a result the topics outlined above should be dealt with in my report.

13. I was instructed that my report should address:
    a. Digital currency technology (as it relates to Bitcoin technology); and
    b. The issues outlined at paragraphs 23-25 of the Re-Re-Amended Particulars of Claim (referred to broadly as "the signing sessions").

14. I have structured my report to deal with these issues in two separate sections, although some of the detail about the way in which digital currency (as it relates to Bitcoin) works was obviously relevant to the issues addressed in relation to the signing sessions.

Documents provided in relation to the technical primer

15. In order to prepare the technical primer, I was provided with the following documents and information:
    a. Re-Amended Particulars of Claim
    b. Re-Amended Defence
    c. Dr Wright's Response to COPA's First RFI
{A/4} d. Re-Amended Reply
    e. Dr Wright's Response to COPA's Second RFI
    f. A transcript to Dr Wright's evidence in proceedings in Norway against Mr. Granath;
    g. Dr Wright's Re-Amended Reply in the Wright v McCormack proceedings
    h. A copy of the blogpost "Jean Paul Sartre, Signing and Significance"
    i. Dr Wright's response to COPA's Third RFI.

Documents provided in relation to my report

16. In order to prepare my report, I was provided with the following further documents and information:
    a. Updated versions of the documents referred to at a. and b. above
{E/1} b. Dr. Craig Wright's First Witness Statement (and documents referred to in that statement)
{E/5} c. Mr. Stefan Matthews' First Witness Statement (and documents referred to in that statement)
{C/5} d. Rory Cellan-Jones' First Witness Statement (and documents referred to in that statement)
    e. Further versions of the "Jean Paul Sartre, Signing and Significance" blogpost described above (which I am told by Bird & Bird were part of Dr Wright's documents disclosed in these proceedings)
    f. The deposition of Mr Gavin Andresen, from the Kleiman v Wright proceedings
{ID_000693} g. Document ID_000693 from Dr Craig Wright's disclosure documents

17. Subsequently, I was informed by Bird & Bird that further documents had been provided by Dr Wright, and these were provided to me and I was asked to consider them:

{ID_004570}
a. ID_004570, which I understand to be PDF copies of emails between Dr Wright and Mr Andresen (that was exhibited to Mr Andresen's deposition in the Kleiman proceedings)

{ID_004572}
b. ID_004572, which I understand to be a PDF print out of an exchange on Reddit that Mr Andresen had with another user (that was exhibited to Mr Andresen's deposition in the Kleiman proceedings)

18. In addition to general consideration of the above materials, I was instructed to consider Dr Wright's first witness statement as it relates to technical matters. I make observations on various parts of Dr Wright's statement throughout my report. However, I was not instructed to (and do not intend to) provide an exhaustive commentary on Dr Wright's statement. {E/1}

19. Furthermore, after starting preparation of my report, I was provided with a copy of Dr Wright's second witness statement, dated 9 October 2023. Dr Wright's second witness statement gives details of his account of what happened during the signing sessions. I have therefore supplemented the parts of the report concerned with the signing sessions to take this material into account. {E/2}

20. In several cases I have disagreed with Dr Wright's technical explanations. In seeking to address the technical matters described by Dr Wright (where appropriate), I have tried to interpret his statements objectively, but it has sometimes been challenging to interpret what is meant where the description is very different to my own understanding of how the technology works. Where I have made assumptions in order to clarify my understanding of what Dr Wright has said in his witness statements, I have tried to identify those assumptions in my report. Where I do not address an issue or parts of either of Dr Wright's witness statements, that does not mean that I necessarily agree with the explanation that Dr Wright has provided.

Bitcoin and various forks

21. There are multiple cryptocurrencies today that include the word "Bitcoin" in their name. The three that are most relevant here are:

    a. The first cryptocurrency is "Bitcoin", originating in 2009. As I mention later in this report, it is also abbreviated to "BTC".

    b. In August 2017, Bitcoin was "forked"[1] to create a new cryptocurrency, derived from the original Bitcoin blockchain, called "Bitcoin Cash" (abbreviated to "BCH").

    c. Then, in November 2018, Bitcoin Cash was forked again, creating a new cryptocurrency called "Bitcoin Satoshi Vision" or "BSV".

22. In this report, I generally refer to Bitcoin as it operated in the period between 2009 and 2014, so it is not necessary to distinguish between these three cryptocurrencies. Where I mention features and terminology that were introduced after Bitcoin was first deployed in 2009, I have explained when I understand the changes to have been made. It is not always possible to be precise about the scope and timing of features and terminology, however, because Bitcoin has been continually developed from its first release in 2009 onwards, changing gradually over time to add new functionality, remove old functionality, fix bugs, and generally maintain the software.

23. It is worth being clear that this type of evolution is inherent in any long-running software project or Internet protocol. As one example, one of the latest RFCs for the TCP/IP protocol says that "TCP is an important transport-layer protocol in the Internet protocol stack, and it has continuously evolved over decades of use and growth of the Internet" (https://datatracker.ietf.org/doc/html/rfc9293 - **Exhibit SM-2**).[2] {H/174}

An overview of Bitcoin

24. Bitcoin is a system for electronic payments originating from the paper "Bitcoin: A Peer-to-Peer Electronic Cash System" by Satoshi Nakamoto (which I refer to in what follows as the "Bitcoin whitepaper"). The paper was published in 2008 and the system was deployed in

[1] I describe the concept of forks in more detail later in this report.

[2] Dr Wright describes TCP/IP in paragraph 131 of his first witness statement as a protocol "which [does] not change." {E/1/25}

2009. Bitcoin is peer-to-peer, meaning users can transfer value between themselves without requiring any single intermediary or central authority.[3] Users transfer ownership of their assets in Bitcoin by forming transactions, in a manner described starting in the "Bitcoin Transactions" section below, which then get verified by other peers in the network.

25. Bitcoin transactions get incorporated into blocks in a process called mining that I describe in "The Role of the Blockchain" section below. These blocks are in turn distributed among and verified by peers, who store them by adding them to a ledger. While different peers might in theory have different copies of the ledger, I describe later in my report how the mining process ensures that all peers will eventually have the same blocks in their ledger. Furthermore, each block added to the ledger includes information in the form of a hash, which is affected by the blocks that were added before it. The ledger is therefore created by linking together individual blocks into a chain, and is known as a blockchain. The contents of one block cannot be changed without changing the contents of every subsequent block.

26. Anyone can download and run the software, and the collection of computers running the software form a peer-to-peer network. This means the ledger is globally distributed and visible to anyone. The unit of account in transactions and in this ledger is a bitcoin.

Cryptographic preliminaries

27. An a*lgorithm* is a sequence of mathematical steps or instructions carried out in a defined order. A *cryptographic primitive* is a set of abstract algorithms that can be run to achieve some purpose, such as encryption. These primitives act as the building blocks when creating more complex systems. Most cryptographic primitives used in real-world applications have a set of standardised implementations, meaning that a standardisation body has assessed various aspects of the collection of algorithms, such as their security, and published them in a form deemed fit for general application.

28. In cryptography, when we say a task is *hard* or *computationally hard*, we mean that the task is assumed to be practically impossible with current technology. This is also known as *intractability*. Specifically, we mean that given state-of-the-art computing equipment we expect it to take a very long time to complete. For example, finding a collision in a hash function (a process that I describe in the "Hash functions" section below) that maintains a 128-bit security level could take 2^{128} steps to find a collision (which in words is over 340

[3] The term "peer-to-peer" is often abbreviated as "P2P".

billion, billion, billion, billion steps). This means that even if each step took only a nanosecond to complete, we would still not expect to find a collision until well past the heat death of the sun.

29. Bitcoin is known as a *cryptocurrency* because it is a cryptographic system, in that it relies on principles of cryptography and uses cryptographic algorithms to form and verify transactions and blocks.[4] The two cryptographic primitives that Bitcoin relies on are *hash functions* and *digital signatures*.

Hash functions

30. A cryptographic hash function is a function, which can be denoted as H, that has the following properties:
    a. It takes in inputs of arbitrary[5] size (meaning that they can be nearly any size and the size does not matter in practice).
    b. It produces an output of some fixed size.
    c. It is efficiently computable, meaning given any input it is fast to compute the output.
    d. It is *pre-image resistant*. This means that given a hash h it is hard to an input x such that H(x) = h.
    e. It is *collision resistant*. This means that it is hard to find two different inputs x and y such that H(x) = H(y).

    The most used hash function today is SHA256, and this is what Bitcoin uses. SHA256 hashes are usually encoded and expressed as a string containing 64 alphanumeric characters, such as the string 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824 (the hex encoding[6] of the SHA256 hash of "hello").

Digital signatures

31. A digital signature is an example of an *asymmetric* or *public-key* cryptographic primitive. This means that it operates using two related keys: a private, or secret, key, and a public key. As the name suggests, the public key can be (and is intended to be) given to anyone. The pair

{E/1/6}

[4] Rather than, as Dr Wright appears to suggest, because it has some tie to anonymity (see Dr Wright's first witness statement at paragraph 22).
[5] In theory there are limits but in practice these limits are extremely unlikely to be reached.
[6] Hex is an encoding scheme that converts groups of binary (0 or 1) values to base 16 values (using the values 0-9 and A-F to represent each *byte*, or group of eight binary digits) instead of more familiar decimal numbers (base 10, using the values 0-9).

of keys is called a *keypair* and the keys have an algebraic relationship; e.g., in many cases the public key is derived deterministically[7] from the private key. It should thus be easy to derive the public key from the private key, but the opposite should be hard: i.e., it must be hard to compute the private key given only the public key.

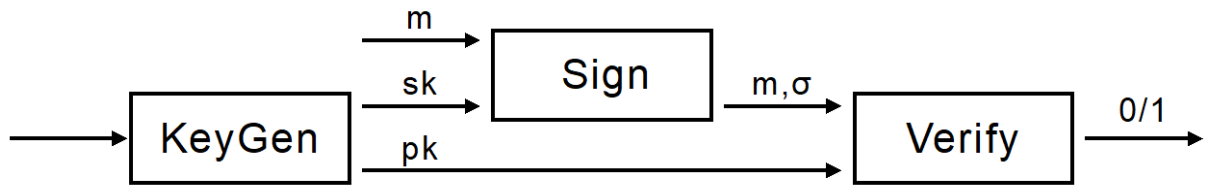


Figure 1. The inputs and outputs of the algorithms KeyGen, Sign, and Verify that make up a digital signature scheme. In terms of notation, m is the message being signed, sk is the secret key, pk is the public key, and σ is the signature.

32. A digital signature acts as the digital analogue of a handwritten signature, in that it is designed to allow an entity to convince others that it has signed a given message. More formally, it is a set of three algorithms denoted as *KeyGen*, *Sign*, and *Verify*. These algorithms behave as follows and as depicted in Figure 1. The inputs and outputs of the algorithms KeyGen, Sign, and Verify that make up a digital signature scheme. In terms of notation, m is the message being signed, sk is the secret key, pk is the public key, and σ is the signature.(and references to the symbols in brackets are the same as the labels given there):

    a. **KeyGen** is a randomised algorithm that produces two keys: a private key (**sk**) and a public key (**pk**). Each time KeyGen is run, it produces a new keypair. As discussed above, these keys have the property that it is hard to compute the private key given only the public key.

    b. **Sign** is a randomised algorithm that allows the holder of the private key to produce a signature (**σ**) on some message (**m**).

    c. **Verify** is a deterministic algorithm allows anyone in possession of the public key to verify that the signer produced a valid signature on a given message. It outputs 0 if the signature does not verify and 1 if it does.

[7] When we refer to an algorithm being "*deterministic*" or its output being "*deterministically*" generated, we mean that if the algorithm is run on the same input multiple times (for example, by multiple different people), the output will always be the same.

33. It should be efficient to run these algorithms, but cryptographically hard to either (1) learn the private key given only the public key or (2) intuitively, sign a new message (and in doing so produce a new signature) without access to the private key. This latter property is often referred to as *unforgeability*. I discuss the security of digital signatures further starting in Paragraph 96.

34. There are several standardised signature schemes, but the one used in Bitcoin is known as ECDSA. This stands for *the Elliptic Curve Digital Signature Algorithm*. As the name suggests, it runs in a particular algebraic setting called an elliptic curve. Curves are defined by a set of parameters, and the curve used in Bitcoin is called secp256k1 and was standardised by an organisation called Certicom. ECDSA signatures are usually encoded and expressed as a string containing 64 alphanumeric characters.[8] The details of how ECDSA works are complicated and not needed to understand Bitcoin further.

35. Like other standardised digital signatures, cryptographers believe that the abstract ECDSA protocol is secure; i.e., that it satisfies unforgeability. This does not mean, however, that there are not insecure implementations or that it is not possible to use ECDSA in an insecure way. In the context of Bitcoin, for example, two researchers (Joachim Breitner and Nadia Heninger) were able to recover hundreds of Bitcoin private keys due to insecure *random number generation*;[9] i.e., due to the fact that the randomness used in forming ECDSA signatures[10] that were published on the Bitcoin blockchain was not generated in a uniform way.

Transacting in Bitcoin

Bitcoin addresses

36. Users in Bitcoin can identify each other in multiple ways. Two examples are:

    a. Users can transact using pay-to-public-key (P2PK) transactions, in which the sender identifies the recipient directly using their full ECDSA public key.

[8] As with a hash, this means they appear in the form of a string of 64 letters and numbers.
[9] Breitner and Heninger, Biased Nonce Sense: Lattice Attacks against Weak ECDSA Signatures in Cryptocurrencies, 2019. Obtained from https://eprint.iacr.org/2019/023.pdf (**Exhibit SM-3)** {H/175}
[10] The fact that the ECDSA signing algorithm is randomised also means it is inaccurate to say that a signature is "unique," as Dr Wright does at paragraph 23 of his first witness statement. {E/1/6}

b. Much more commonly, users transact using *addresses*, which are alphanumeric identifiers (i.e., strings consisting of letters and numbers) that are different from (but often related to) the full public keys, as I explain below.

37. In the simplest case, an address is (deterministically) derived from either an existing or a newly generated ECDSA public key. Generating an ECDSA keypair is highly efficient, which means that it can be done by a computer easily. This means that a user can generate and use a different address every time they transact if they do not wish to re-use an address.

38. This simplest type of address is referred to as pay-to-public-key-hash, or P2PKH for short. There are new address types that were introduced starting in 2012, such as pay-to-script-hash (P2SH) and Bech32. It is typically possible to distinguish between these different address types by looking at the first characters in the address, which indicates the type of the address (e.g., P2PKH addresses start with 1 and P2SH addresses start with 3). For simplicity, and because they were the only type of address before 2012, I use P2PKH addresses in my examples and discussions below.

39. When addresses are derived from public keys, each address has an associated private key that remains secret as long as the user keeps it secret. The private key can be used to sign messages, whose connection to the address can then be verified. In other words, given an address, a public key, a signature, and a message, anyone can verify whether or not (1) the address was derived from the public key, by re-running the derivation algorithm for the public key and confirming whether or not the resulting address matches the expected address, and (2) the signature and signed message are valid for that public key (by running the signature verification algorithm). These properties are used in Bitcoin to allow users to transfer ownership of bitcoins they possess, in a manner that can be independently verified by anyone (but that does not require the establishment of a user's real-world identity).

Bitcoin transactions

40. In Bitcoin, a transaction can have multiple senders and recipients. Senders and recipients are identified using addresses, and the value being sent or received by each party is identified in bitcoins. In a similar way that dollars and pounds are divisible into cents and pence, bitcoins are divisible, and can be divided to the eighth decimal place; i.e., the smallest amount it is possible to send is 1×10^{-8} bitcoin (0.00000001).

41. A unit of bitcoin is abbreviated as "BTC". I don't know exactly when this terminology was introduced, but early threads on the popular Bitcointalk forum suggest that it was already in use in December 2009 (https://bitcointalk.org/index.php?topic=15.0 (**Exhibit SM-4)** and https://bitcointalk.org/index.php?topic=16.0 – **Exhibit SM-5**). {H/176} {H/177}

42. The unit of the smallest division, 1 x 10^{-8} BTC, is termed a "s*atoshi"* or "*sat"*, named after the creator of Bitcoin. This terminology was not used when Bitcoin was first released in 2009. As far as I can tell, the term "satoshi" was first proposed for this usage by a user on Bitcointalk in February 2011 (after first proposing it to mean 0.01 BTC in November 2010).[11]

43. To describe how transactions work, I start with the simplest example in which there is one sender and one recipient, each identified according to a single address. Such a transaction includes one input (corresponding to the sender) and one output (corresponding to the recipient). We can consider the *transaction output* (abbreviated as *TXO*) as consisting of the recipient's address and the value in BTC that is being sent to that address. I will refine the definition of the *transaction input* later, but for now it can be thought of as analogous to a TXO, namely as containing the sender's address and the value in BTC being sent by the address. To ensure that the sender, as the current owner of the bitcoins, is really the one authorising their transfer, a Bitcoin transaction also needs to contain a digital signature from the sender, where the message being signed contains the rest of the information detailing the transaction. A transaction input thus also includes the public key corresponding to the sender address and a signature by the sender over the rest of the information in the transaction (e.g., the list of TXOs).

44. Given this information, other peers in the network can verify the transaction using the two properties described in Paragraph 39: they can look at the address, public key, and signature in the transaction input and check that the public key corresponds to the address and that the signature verifies. Since transactions are public, it is also possible to check whether or not this address was used in any TXO before; i.e., to confirm that this address did in fact receive the number of bitcoins it is spending in the past. This allows peers in the network to determine if bitcoins are being transferred by their current owner or not, but fails to detect an attack called *double spending*, in which the current owner of some bitcoins might try to spend them multiple times. The ability to detect and prevent double spending is a necessary requirement for any payment system. To prevent double spending, Bitcoin keeps track of

[11] This user summarises this history, with relevant pointers, at https://bitcointalk.org/index.php?topic=407442.msg4415850#msg4415850 (**Exhibit SM-6**) {H/178}

which transaction outputs are *unspent* and allows only these unspent outputs to spend the coins they received. This means that Bitcoin has the property that all bitcoins received in a transaction need to be spent all at once. It also means that a transaction input does not explicitly specify the BTC value to be transferred, but rather refers to a previous unspent output (which contains the value already).

45. Going forward, I refer to an unspent transaction output as a *UTXO*. This term was not used in the Bitcoin whitepaper or in the original codebase when Bitcoin was released in 2009, but was introduced sometime in 2011 or 2012 and is now used commonly in the context of Bitcoin and cryptocurrencies more generally (see, e.g., {H/179} https://en.wikipedia.org/wiki/Unspent_transaction_output (**Exhibit SM-7**). For example, it is now common to refer to Bitcoin as operating in a "UTXO model" as opposed to the "account model" used by other cryptocurrencies such as Ethereum.

46. Moving beyond the simple example with one input and one output, transactions with multiple inputs function in the same way: each transaction input needs to specify its own distinct UTXO and valid signature on the transaction data. Transactions with multiple inputs do not necessarily have multiple senders, as they could just represent one sender spending the contents of multiple UTXOs associated with the same address.

47. Similarly, transactions can have multiple transaction outputs, where again there can be multiple distinct addresses (representing different recipients) or not. This latter type of transaction is needed to divide bitcoins, as again any bitcoins received in a transaction must be spent all at once. For example, if a user has 10 BTC associated with a UTXO and wants to send two bitcoins to another user, they can form a transaction with one input representing their 10 BTC UTXO (and a valid associated signature) and two outputs: one containing the address of the other user and receiving 2 BTC, and the other containing an address they control and receiving 8 BTC. In this way, a user can *make change*, just as happens when spending physical cash.

48. In the screenshot of a sample transaction in Figure 2 below, we can see a transaction that has one input and two outputs. The two outputs are the addresses 16M8x…3gL (receiving 1.85451366 BTC) and 1DKLE…M3M (receiving 505.59095385 BTC). The input is the transaction hash 53305…43b representing the transaction in which this input received 507.44546751 BTC (by acting as a TXO).

Figure 2. The basic information of a Bitcoin transaction with one input (depicted in the box on the left) and two outputs (depicted in the boxes on the right). The string in the top left is the transaction hash. This and later screenshots of transactions and blocks were taken from btcscan.org; this specific transaction can be found at {H/180} *https://btcscan.org/tx/660db6149ec8f6daa1b4ba0ec117a3a550a75bbcd8b60ad4ac028339af03da23 - **Exhibit SM-8.** This website is an example of a block explorer; i.e., a third-party source for looking through the Bitcoin blockchain.*

49. In this transaction, the sum of the outputs is exactly equal to the input value. In general, a transaction is valid as long as the sum of the output values is less than or equal to the sum of the input values. In the case that the output sum is strictly less than the input sum, the remainder is left as a *transaction fee*, which gets paid to the special participant called a *miner* who acts to seal this transaction in the final ledger. I explain the role of this participant in the next section. The transaction fee can be set manually by the user or automatically by the software. Unlike the block reward (which is fixed, as I discuss below), the transaction fee is optional and acts as an incentive for the miner to support the network.

50. The above paragraphs describe the most basic type of transfer operation in the Bitcoin network, in which one user wants to transfer ownership of their bitcoins to another, using the address of the recipient to identify them. Transactions in Bitcoin are represented using a programming language called *Script*, however, that can support more advanced functionalities. Like other low-level programming languages, Script consists of a set of instructions, which in Bitcoin are called *opcodes*. These instructions can represent basic numeric operations such as adding two numbers (OP_ADD) or a check that two numbers are equal (OP_NUMEQUAL). They can also express more advanced cryptographic operations such as performing a SHA256 hash (OP_SHA256) or checking that a signature is valid for a given public key and message (OP_CHECKSIG).

51. The full details of how Script works and the functionalities it supports are out of the scope of this report. As one example, however, we continue with the simple P2PKH transaction used in Figure 2 above. The full details of this transaction, including the Script instructions for both the input and the outputs, are in Figure 3.

Figure 3. More detailed information of the same Bitcoin transaction as above. Notably, we can see the Bitcoin Script used, the address used in the input, and the transactions in which the outputs later spent their contents.

52. In this transaction, both outputs have the same set of instructions in the "scriptPubKey (ASM)" field: OP_DUP OP_HASH160 OP_PUSHBYTES_20 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG. The value provided for pubKeyHash is the hash of the recipient's public key hash using a standardised hash function called RIPEMD-160. Since it is deterministic, it should thus match (according to OP_EQUALVERIFY) the RIPEMD-160 hash (performed using OP_HASH160) of the "scriptPubKey (hex)" field in that output. The OP_CHECKSIG instruction means that the transaction must contain a valid signature, where the message being signed is derived from the transaction, the signature is given as input in the "scriptSig (hex)" field, and the public key is taken from the "scriptPubKey (hex)" field in the TXO of the transaction in which this input received bitcoins.

53. The above descriptions have focused on P2PKH addresses, and above I briefly mentioned other more advanced address types. As described previously, an even simpler transaction script is pay-to-public-key (P2PK), which specifies the recipient directly by a public key rather than by an address. This means that the receiving entity does not need to later provide the public key as input when spending this TXO, as it is already embedded in it. Despite the simplicity of this type of transaction script, it has been used much less extensively than

P2PKH,[12] despite the fact that both were supported by the original version of the Bitcoin software. One possible explanation is that using P2PKH is more efficient in terms of the information a transaction recipient has to provide to the sender, as an uncompressed ECDSA public key is 65 bytes whereas an address is only 25 bytes.

54. One consequence of the UTXO model is that, given access to every Bitcoin transaction that has taken place (which, as discussed next, is the role of the blockchain) it is possible to follow the entire history of a bitcoin from the point at which it was generated (as described in Paragraph 60) to the current UTXO where it is held. In other words, transactions form a chain, with each transaction pointing backwards to the transaction(s) in which its input(s) acted as a TXO. This gives rise to the ability to trace the flow of bitcoins by following this chain forwards or backwards.

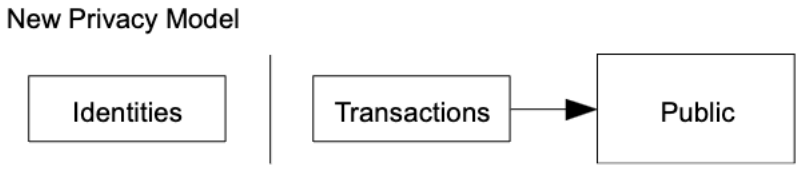


Figure 4. An image from the Bitcoin whitepaper demonstrating the separation between (private) identities and (public) transactions.

55. Crucially, however, Bitcoin still achieves some level of anonymity, or rather pseudo-anonymity (or pseudonymity), by identifying users in transactions using (ephemeral) addresses derived from ECDSA public keys as opposed to real-world identities. Section 10 of the Bitcoin whitepaper states that "*privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone."* This is also depicted in an image in the whitepaper in which identities are kept separate from transactions, copied here as Figure 4. I raise this point only because of the (in my opinion, erroneous) claim in Dr Wright's first witness statement at paragraph 17 that one "of the crucial elements [of Bitcoin] is traceability and identity, which is the system's ability to track transactions and ultimately identify the parties behind them." It is worth mentioning, however, that the anonymity guarantee provided by Bitcoin has been shown to be fairly weak in practice.

[12] There are nevertheless some prominent examples of P2PK transactions, such as the coin generation transaction contained in the genesis block (described in "The Role of the Blockchain" section and depicted in Figure 5 below).

56. Dr Wright's first witness statement also states that nodes are "de-anonymised if they acquire significant power over the network" (paragraph 14). It is true that some degree of de-anonymisation may be possible if (1) a miner uses a consistent IP address or uses a consistent Bitcoin address to collect the block reward or (2) uses their power (assuming it is greater than 51% of the mining power) to carry out transactions that make it clear which other Bitcoin addresses they own; e.g., by performing double spending from a specific set of addresses. These behaviours could just as easily not take place, however, and furthermore do not have anything to do with the hash puzzle competition itself.

The Role of the Blockchain

Transaction ordering

57. In checking transaction validity, I described above how it is important for peers to be able to check for double spending. Doing so requires a peer to know whether or not a transaction has been seen by other peers, and furthermore which transaction they saw before another. To illustrate why, consider the case in which two peers ("**P1**" and "**P2**") start with the same list of UTXOs containing bitcoins owned by a participant **A** and are both observing transactions on the network.

    a. If **P1** sees a transaction sending those bitcoins from A to participant **B**, then P1 will accept the transaction as valid,[13] update A's TXO with B's on its own UTXO list, and thus reject any future transactions with A's TXO as input.

    b. If **P2** sees a different transaction with A's same TXO as input, but with an output TXO belonging to a different participant **C** instead, then P2 will update its list differently to P1.

    c. P1 and P2 would then end up with different UTXO lists: P2 would list C as the owner of those bitcoins, while P1 would record B as the owner.

58. Resolving this problem requires peers to share information, but in a peer-to-peer network it is inevitable that different peers will see transactions at different times. A global notion of *transaction ordering* across peers is therefore necessary. This is the role played by the Bitcoin blockchain, which acts as the ledger of all valid transactions that have propagated

[13] Assuming the signature and other transaction data passes verification.

through the network. Transactions are ordered in the ledger by being formed into *blocks*, and having each block refer to the previous one so that they form an interlinked sequence called a *blockchain*. This defines an order, because all the transactions in the previous block can be seen as having taken place before all the ones in the current one. Blocks thus have a *height* in the blockchain, with again all transactions in the block at height H1 taking place before the transactions in the block at any higher height H2.

Genesis block

59. The first block in the blockchain (at height 0) is called the *genesis block*. This block is hardcoded into the Bitcoin software, thus giving any participant wishing to engage in mining an initial block to chain back to. The block was produced on 3rd January 2009 at 18:15:05 UTC (according to its timestamp) and contains a single coin generation transaction; the purpose and form of these types of coin generation transactions is defined below. The transaction is shown in Figure 5, in which we can see that the script used as input to the transaction contains information (again encoded using hex). This information is often called the *coinbase message,* and in this case when it is decoded it reads: *"The Times 03/Jan/2009 Chancellor on brink of second bailout for banks".* This is generally understood as a reference to the headline for The Times newspaper on that date, 3rd January 2009, thus proving that this block could have been created only on or after this date.

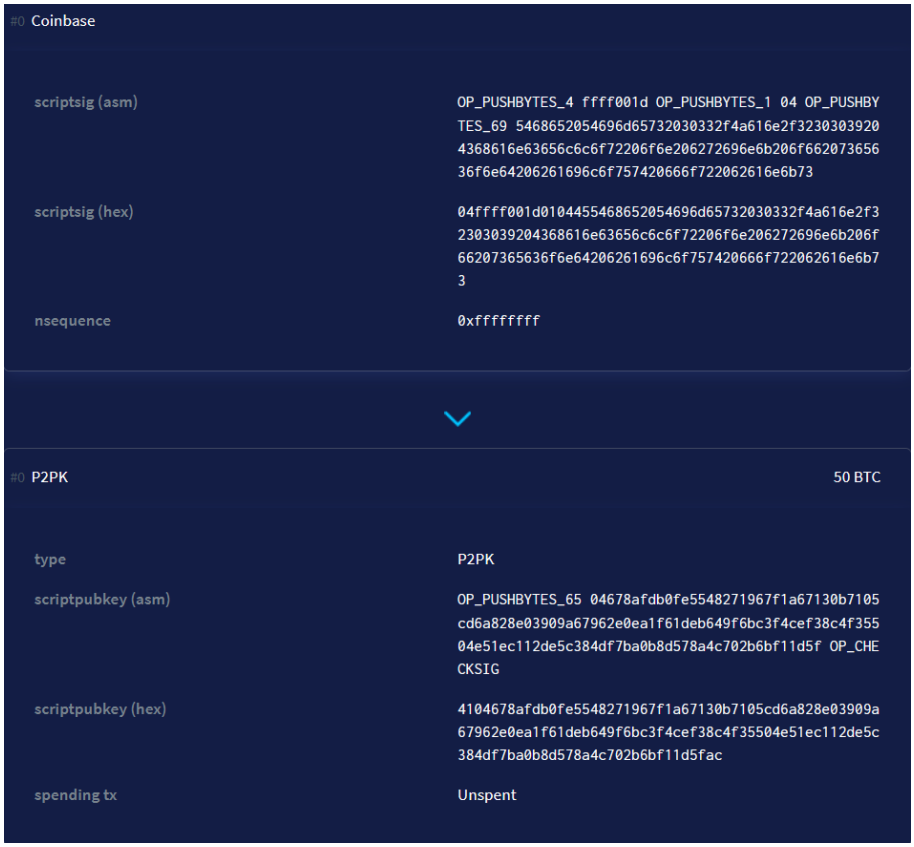


Figure 5. The coin generation transaction contained in the genesis block. This screenshot was taken from https://btcscan.org/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f - ***Exhibit SM-9***

{H/181}

60. Following the genesis block, individual blocks are produced as follows. First, a participant collects recent valid transactions that have been broadcast in the network into a list. They then add at the start of this list a special *coin generation* transaction that sends a specific number of bitcoins from no input to an output address that they control (i.e., for which they know the private key). This coin generation transaction acts to create new bitcoins, which is why this process is referred to as *mining* and participants that perform this are referred to as *miners*. The newly mined bitcoins, known as the *block reward*, act to incentivise the miner to perform the computation necessary to produce a block. The coin generation transaction also guarantees that each miner competing to form a block will do so with a unique set of transactions.[14]

[14] Unlike Dr Wright (cf his first statement, paragraph 12), I differentiate between the roles of a **miner**, which competes to form blocks, and a **(full) node**, which validates transactions, blocks, and the overall blockchain in a way that is non-competitive and involves the receipt and processing of broadcast information rather than the monitoring of other nodes. This differentiation is also not made explicitly by the Bitcoin whitepaper but is useful to describe the different actions taken in the two roles, even if they end up being performed by the same participant.

61. After including the coin generation in the list of transactions, the miner then hashes that list, using a specific data structure called a *Merkle tree*,[15] which forms a *Merkle hash* that represents all the transactions. The miner then inputs this Merkle hash into a hash function, along with fields such as the hash of the previous block and a *nonce.* A nonce is a cryptographic number (or string of characters) created to be used only once, so that the output is unique to that usage.[16] The nonce is not predetermined and is chosen by the miner. In this case, we can think of the nonce as being set initially to 0. If the hash produced by this computation is below a specific *target*, which we can think of as being a hash starting with a specific number of zeroes, then the miner has produced a valid block. Not all nonces will result in that target being met, and if not, the miner continues trying to produce a block by changing the value of the nonce (we can think of them as just adding 1 to it) and computing the hash again. The nonce is necessary for this process as the hash function is deterministic, meaning it always produces the same output given the same set of inputs: the use of a nonce thus introduces variability into the output, resulting in each block having a different hash.

62. If we think of the hash function as producing outputs that are distributed randomly, and think of the binary representation of a hash output, then if we need the hash to start with a single zero we have a 1/2 chance of this happening (since the hash can start with either a 0 or a 1). This means that the average number of times we would have to perform the hash (with a different nonce each time) would be two; the number for a given block might be higher or lower, but no matter what we would be very likely to find such a hash quickly. If we need the hash to start with two zeroes, we have a 1/4 chance of this happening, so on average we would find such a hash after four computations. In general, if we need the hash to start with N zeroes, we have a $\frac{1}{2}^N$ chance of this happening, which means we need to perform 2^N computations on average. For larger values of N this is a significant amount of computation, or *work*, which is why this process run by miners is known as *proof of work*.

63. Proof of work is not unique to Bitcoin. The first proposal of proof of work was in 1992, by Cynthia Dwork and Moni Naor, and suggested requiring a proof of work when sending an email to combat the problem of spam. The specific type of proof of work used by Bitcoin is derived from a previous proposal called *Hashcash*, as proposed by Adam Back in 2002 and is

[15] Merkle trees are not necessarily balanced. In fact, a very common way to implement them is as a so-called history tree (also called a Merkle mountain range in the blockchain community), in which the left subtree can be significantly larger than the right. I make this observation to correct matters as stated in Dr Wright's first witness statement at paragraph 20, with which I disagree. {E/1/6}

[16] It is sometimes considered to be an abbreviation of a "number used only once".

referred to in the Bitcoin whitepaper. More generally, many of the building blocks used in Bitcoin were developed in the 1980s and 1990s, as is laid out in an article from 2017 by Arvind Narayanan and Jeremy Clark (which can be found at
{H/182} https://queue.acm.org/detail.cfm?id=3136559 – and a copy is exhibited at **Exhibit SM-10**). I briefly summarise some of the most relevant building blocks here:[17]

a. In terms of transactions, the idea of identifying users using pseudonyms that are derived from public signing keys was present in the original proposal for electronic cash by David Chaum in 1981, and was also used in the b-money protocol proposed by Wei Dai (in 1998)[18] and the bit gold proposal by Nick Szabo (in 2005).[19] The idea of linking transactions to form a chained history of a coin's owners is implicit in the bit gold proposal; e.g., it suggests that "to verify that Alice is the owner of a particular string of bit gold, Bob checks the unforgeable chain of title in the bit gold title registry."

b. In terms of generating new coins, both b-money and bit gold propose using proof-of-work for this purpose; e.g., b-money states that "anyone can create money by broadcasting the solution to a previously unsolved computational problem. The only conditions are that it must be easy to determine how much computing effort it took to solve the problem and the solution must otherwise have no value, either practical or intellectual."

c. In terms of creating the ledger, the idea of having individual blocks of data that are linked by having each block point to the previous one (and in particular using this to instantiate a timestamping service) was first proposed by Stuart Haber and W. Scott Stornetta in 1990. This is also relied on in the "distributed property title registry" used in bit gold. Furthermore, the idea of grouping together transactions in a block using a Merkle tree is contained in later papers written by Haber and Stornetta.

64. Proof of work secures blocks in the blockchain because after the miner produces a block, they broadcast it into the network. Once other miners receive it and verify that the block is valid,

---

[17] Dr Wright refers to his experience implementing secure logging at Vodafone as a "significant steppingstone on the path to creating Bitcoin" (in paragraphs 45-47 of his first witness statement), but secure logging has been a standard recommended practice in the IT industry for decades; see, for example, the following guidance published by NIST in 1996: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-14.pdf - **Exhibit SM-11**
{H/183}

[18] Found at http://www.weidai.com/bmoney.txt - **Exhibit SM-12**
{H/184}

[19] Found at https://unenumerated.blogspot.com/2005/12/bit-gold.html - **Exhibit SM-13**
{H/185}

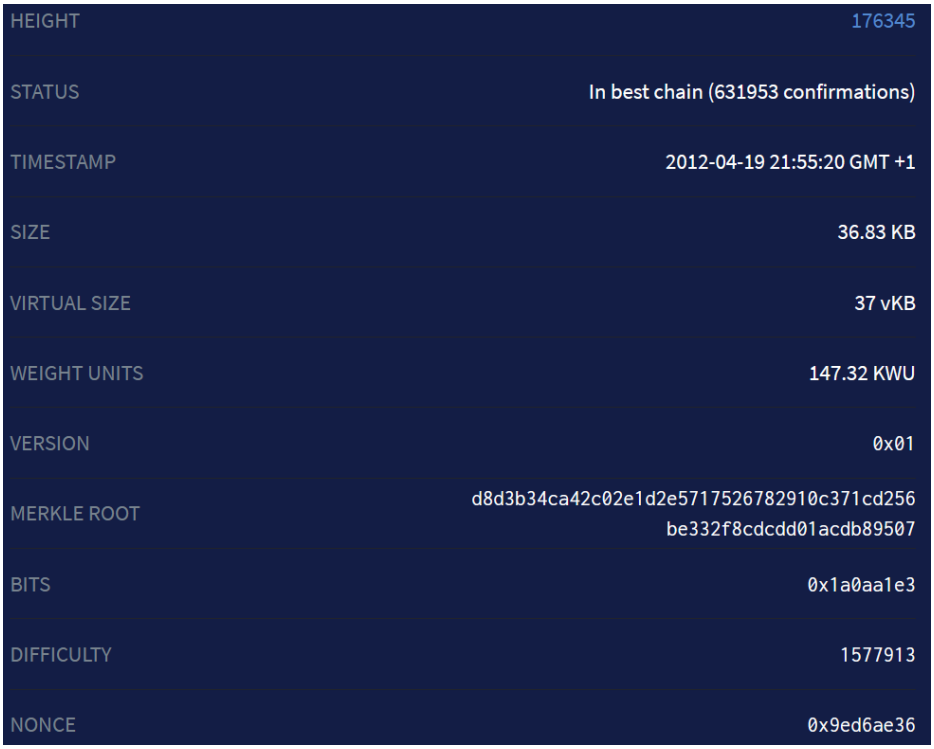
they can then continue mining on top of this block by using its hash (as the previous block hash needed in their own computation). If producing blocks were computationally easy, however, then miners might have many options as to which previous block to use, with no clear way to pick one. Worse, if they picked a previous block based on some kind of majority opinion (e.g., the majority of peers they heard from proposed it as a block), then they could be subjected to a *Sybil attack*, in which an attacker who wants to promote one particular block could run a huge number of participating nodes in the network and instruct them to all vote in the same way, thereby skewing the vote. Using proof of work thus increases the computational cost of launching such an attack.

65. Proof of work makes it computationally difficult to produce a block, but it may still be the case that two valid blocks are produced at the same time. Miners that receive these blocks must thus choose which of the two to use as their previous block. Miners should always extend the chain with the most cumulative work; i.e., they should always pick the sequence of blocks (starting at the genesis block and including the latest one) that was harder to produce. There thus may be temporary inconsistencies in what is sometimes called the *tip* of the blockchain (meaning the blocks at the highest heights), but given the above *fork-choice rule* (i.e., that the valid chain is the one with the most cumulative work), over time all peers in the network will agree on the state of the ledger.[20] Blocks that have been mined and are not part of this chain are sometimes called *orphans* and the miners who produced them do not get any reward.[21] The number of blocks that have come after the block of a given transaction is referred to as that transaction's number of *confirmations*, with a higher number of confirmations giving greater confidence that that transaction is included in the canonical ledger.

[20] The exception is if one or multiple peers have been isolated from the rest of the network, or if there is a network partition, meaning certain sets of peers do not communicate with others so might independently extend the blockchain without being aware of the blocks produced by others. This type of *eclipse attack* is considered prohibitively expensive to sustain for any meaningful period of time, so Bitcoin does achieve the property of *eventual consistency*, meaning all peers eventually agree on the state of the ledger.

[21] In other words, a block could be accepted as valid, and even mined from, without necessarily ending up as part of the longest chain (the "Bitcoin blockchain") if another sequence of blocks ends up having a higher cumulative difficulty (cf Dr Wright's first witness statement, paragraph 14).

{E/1/4}



HEIGHT	176345
STATUS	In best chain (631953 confirmations)
TIMESTAMP	2012-04-19 21:55:20 GMT +1
SIZE	36.83 KB
VIRTUAL SIZE	37 vKB
WEIGHT UNITS	147.32 KWU
VERSION	0x01
MERKLE ROOT	d8d3b34ca42c02e1d2e5717526782910c371cd256 be332f8cdcdd01acdb89507
BITS	0x1a0aa1e3
DIFFICULTY	1577913
NONCE	0x9ed6ae36

Figure 6. The header for the block at height 176,345. This information was taken from btcscan.org; this block can be found at https://btcscan.org/block/000000000000007f68824b732fcb11e5c04790c859a99e2415f05bc68766bcb45 - {H/186}

Exhibit SM-14

66. Blocks are split into the *header*, which contains high-level information about the block, and the data consisting of all the transactions in the block. **Error! Reference source not found.**Figure 6 above shows the header for the block at height 176,345. The third field is the timestamp, which represents the time at which the miner produced the block (according to their own local clock). This field provides an additional way to modify the inputs to the hash function, and other peers will receive the block later according to the time it takes to relay the block across the network. The other important fields to consider are at the end: the *Merkle root* shows the Merkle hash of the transactions contained in the block. The *difficulty* defines the target hash that a block's hash needs to be strictly under. Finally, the nonce is the value that the miner selects when doing the work necessary to create the block as described above: the specific nonce listed here, the hex-encoded number 0x9ed6ae36, is the one that allowed the miner of this block to have the block's hash be under the target. The hash of the previous block (i.e., the block at height 176,344) is not pictured here, but it is also given as input to the block's hash computation and is needed to ensure that the blocks form a chain.

67. The blockchain can only ever grow in size, as new blocks and transactions get added but never removed. The Bitcoin blockchain currently requires more than 450 GB to store. As

such, some peers in the network choose to store block headers but not every transaction. These peers are called *light clients*, or *SPV* clients (short for *Simplified Payment Verification*; this is the original terminology that was used in Section 8 of the Bitcoin whitepaper). While it is still possible to perform some types of verification and to perform and relay individual transactions as a light client, checking fully that new transactions comply with the consensus rules (and in particular that they are not double spending) requires storing, at a minimum, the full list of UTXOs (as observed in Section 7 of the whitepaper, this "condensed" history suffices in place of the full list of all transactions). Peers that store this information and perform a full set of checks for all new blocks and transactions are called *full nodes*.[22]

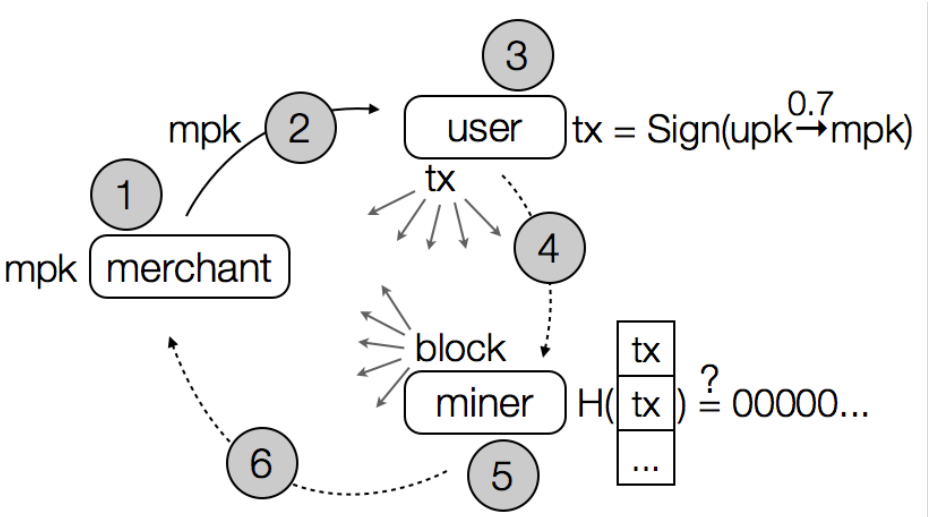


Figure 7 (taken from "A Fistful of Bitcoins: Characterizing Payments Among Men with No Names" by Meiklejohn et al., Communications of the ACM, April 2016). An overview of how Bitcoin works – ***Exhibit SM-16***

{H/188}

68. Figure 7**Error! Reference source not found.** presents a summary of how Bitcoin works, using as a hypothetical example the case in which a user wants to pay a merchant 0.7 BTC in exchange for some goods or service. In Step 1, a merchant either generates a new signing keypair (and thus a new address) or selects an existing address, denoted in either case **mpk**. In Step 2, they send the address to the user. In Step 3, the user forms a transaction transferring ownership of 0.7 BTC from a UTXO associated with one of their own addresses, denoted **upk**, to the merchant's address **mpk**. This transaction includes a cryptographic signature so that other parties can verify that it really is the user in possession of the private key corresponding to **upk** that is transferring ownership. Other parties perform exactly this verification when the user broadcasts the transaction into the network in Step 4, forwarding it

{H/187}

[22] It is not possible to know the exact number of full nodes running in Bitcoin today, but https://bitnodes.io/ estimates the number of reachable nodes to be 17,070 (as of October 2023) – see **Exhibit SM-15.**

on to their peers if it passes verification. Eventually, the transaction reaches a miner, who in Step 5 seals the transaction into a valid block; i.e., a block whose contents produce a hash of the appropriate difficulty and meets all other consensus rules (e.g., does not contain any double spending). As the user did with their transaction, the miner broadcasts this block into the network, and peers check its validity by checking the proof of work. Once the block has been included at a sufficient depth in the blockchain (i.e., once the transaction has enough confirmations), the merchant can be convinced that the payment has gone through and provide the user with their goods or service.

69. There are two values in the descriptions above that are parameters of the system: the number of coins a miner creates in a coin generation transaction, and the target that a block's hash needs to be below. These parameters both change over time:

    a. For the block reward, Bitcoin was designed to generate 21 million bitcoins in total.[23]
    b. The initial block reward was 50 BTC.
    c. The block reward halves every 210,000 blocks.
    d. It first halved in November 2012 to 25 BTC.
    e. It halved again in July 2016, to 12.5 BTC
    f. It halved again in May 2020 to 6.25 BTC, which remains the current block reward as of September 2023.
    g. It will eventually drop to 0, at which point miners will earn rewards only through transaction fees.

70. Figure 8 shows the number of bitcoins that have been generated over time, with the three inflection points (visible as sharp corners on the curve) showing the points at which the block reward halved.

[23] The limit is technically just under 21 million, because the software was designed in a way that makes it impossible to spend the block reward from the genesis block.

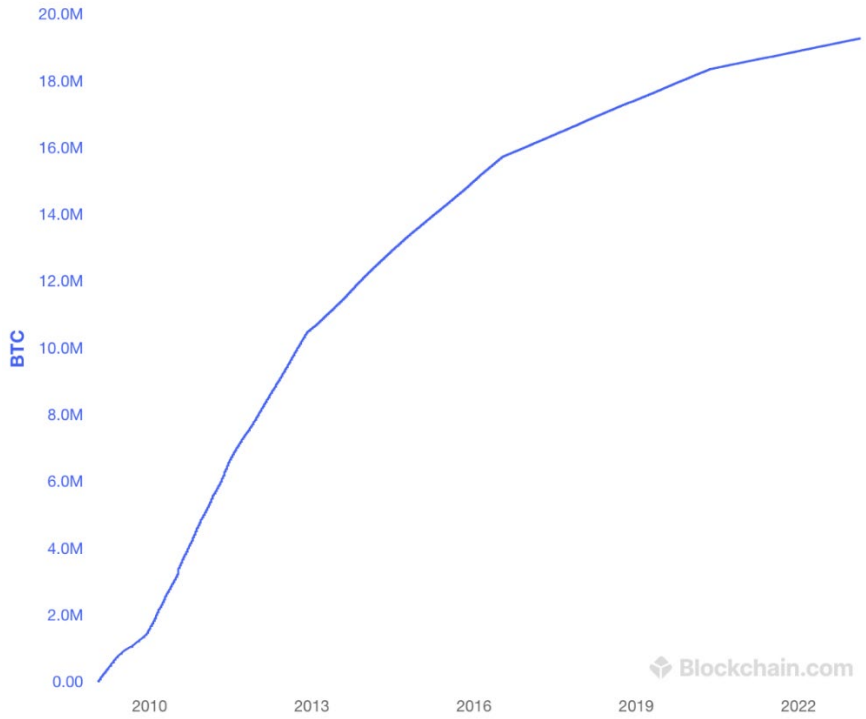


Figure 8. The number of bitcoins that have been generated over time. This graph was taken from {H/189} *https://www.blockchain.com/explorer/charts/total-bitcoins* - ***Exhibit SM-17.***

71. For the target hash, Bitcoin is configured to have a new block produced every 10 minutes on average. This means that the target needs to change according to the collective computational power of the peers participating and competing in the mining process. New miners add to the computational power of the network and thus make it faster to find a block with a fixed leading number of zeroes; conversely, any reduction in the computational power of an existing miner makes it slower. Figure 9 shows how much the network's computational power has changed over time, in terms of the number of hashes that get computed per second. In February 2011, the network was already computing 402 billion hashes per second (0.402 terahashes per second (TH/s)). In February 2023, it was computing 313,660,575.839 TH/s, meaning the hash rate was more than 780 million times higher in February 2023 than it was in February 2011. Figure 10 shows how the target hash has thus gotten lower, meaning mining has gotten more difficult, over time (although we also can see that there have been times in which the difficulty has decreased due to a reduction in the collective hash rate of the network). The difficulty graph follows a similar trend to the hash rate graph, reflecting how the difficulty (and thus target hash) gets recalculated approximately every two weeks to reflect the changing hash rate of the network. Thus, whereas the formula for computing the difficulty is pre-defined in the Bitcoin software, the difficulty itself changes according to the expected time to produce 2016 blocks divided by the actual time, meaning if it takes less time

than expected the difficulty goes up and if it takes more time than expected the difficulty goes down.

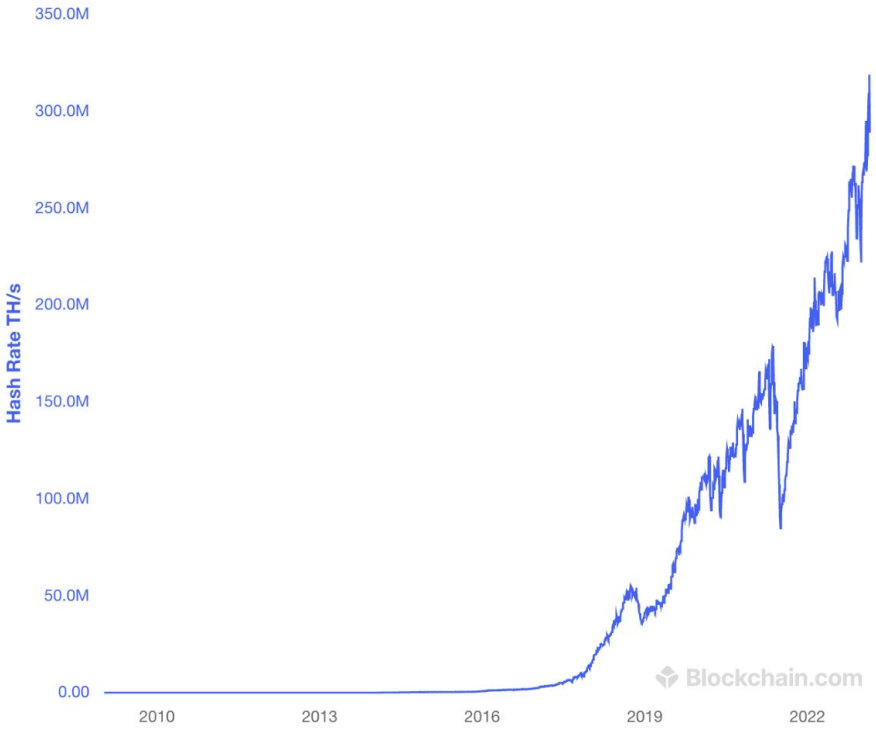


Figure 9. The collective hash rate in the Bitcoin network, in terms of the number of hashes that get computed per second (measured in terahashes, where 1 terahash = 1 x 10^{12} hashes). This graph was

{H/190} taken from https://www.blockchain.com/explorer/charts/hash-rate - **(Exhibit SM-18)**.

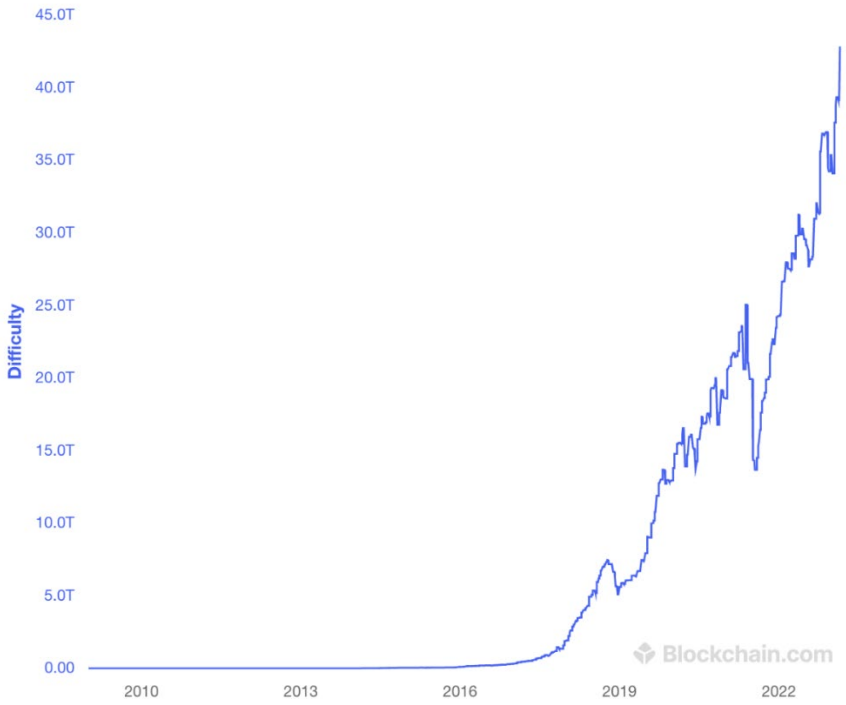


Figure 10. The difficulty over time. As the difficulty gets higher, the target hash (i.e., the hash that block hashes must be below) gets lower. This graph was taken from {H/191} *https://www.blockchain.com/explorer/charts/difficulty - **(Exhibit SM-19)**.*

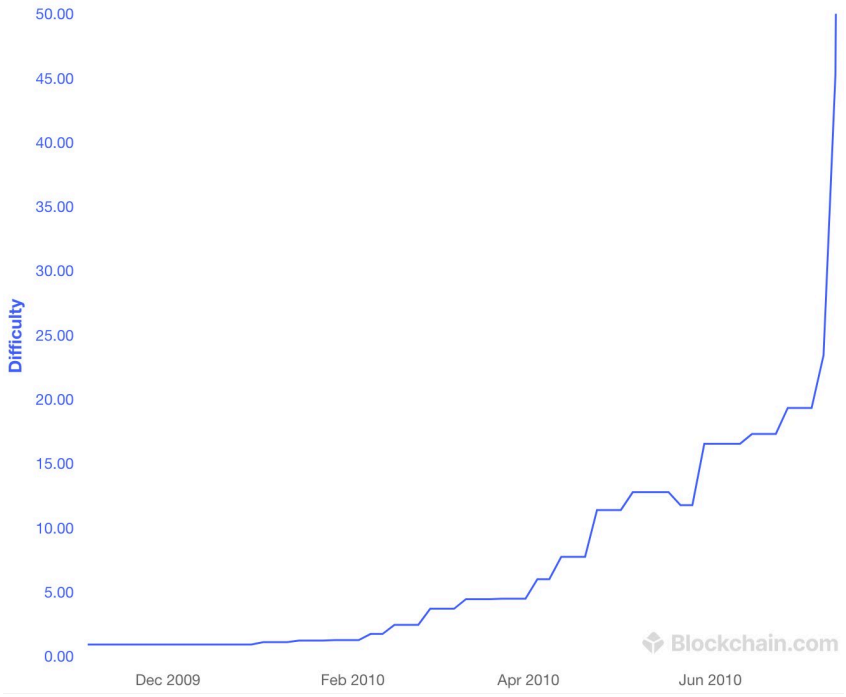


Figure 11. The difficulty between January 2009 and August 2010.

72. One consequence of this increased hash rate and difficulty is that it is no longer feasible to mine bitcoins with a personal computer; at the current difficulty, we would expect it to take a laptop computer tens or hundreds of millions of years to be the first to find a valid proof-of-work. Today, individual users can share resources and join a *mining pool* in which they contribute their computational power so that, if the pool is successful in mining a block, the reward gets distributed among the users in some proportion. Bitcoin is also mined commercially in devoted datacentres, which consist of specialised computer chips (known as application-specific integrated circuits, or ASICs) created solely for the task of mining bitcoins. This has led to the current situation, depicted in Figure 12, in which a small number of pools are responsible for mining the majority of blocks in the Bitcoin blockchain (yet difficulty is the highest it has ever been).

73. As illustrated in Figure 11, however, the difficulty was 1 throughout 2009, going above 1 for the first time in January 2010. This difficulty "corresponded to 4-8 general-purpose cores running the nonce-search algorithm, trying out ~7M double-SHA hashes per second;"[24] in other words, the collective hash rate was roughly 7 megahashes/second, which could have been produced by 4-8 laptop computers at the time.

74. Bird & Bird has asked me about the comments made by Dr Wright on his mining setup (in {E/1/29} paragraphs 115-117 of his first witness statement). In my opinion it would not have been necessary to run a setup of that magnitude to mine Bitcoin at that time, and in fact it could not have been running in 2009 or early 2010 as if it had it would have increased the difficulty considerably (in contrast to what was observed, as discussed above).

[24] This quote is taken from "Bitcoin and The Age of Bespoke Silicon" by Michael Bedford Taylor, published in 2013 and available at https://www.bsg.ai/papers/bitcoin_taylor_cases_2013.pdf. **(Exhibit SM-20)** These numbers are also supported by a Bitcointalk thread from 2010, https://bitcointalk.org/index.php?topic=1628.0, showing the hashrate of different CPUs at the time (**Exhibit SM-21**).

{H/192}

{H/193}

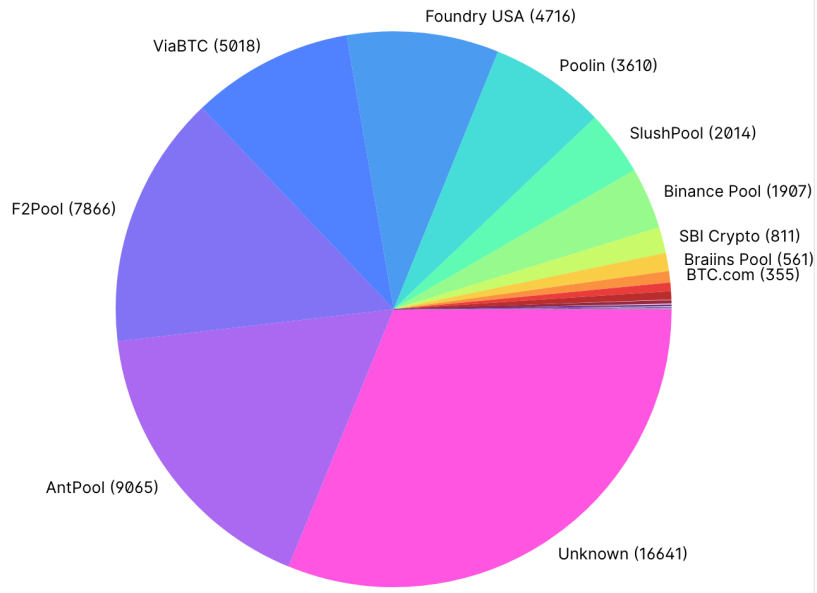


Figure 12. From February 2022 to February 2023, the number of blocks mined by different mining pools. This screenshot was taken from https://www.blockchain.com/explorer/charts/pools.

75. Beyond changing over time, both the payment of a block reward and target hash are defined by consensus. In other words, there is nothing in the protocol itself that stops miners from using any values they want; e.g., they could send themselves 100 BTC or produce an "easy" block whose hash has a small number of leading zeroes. If they do not align with what the rest of the network considers acceptable, however, then other nodes will just reject their block and it will not become part of the consensus view of the Bitcoin blockchain. In this way, the users of Bitcoin form a collective agreement of the rules they choose to follow. Bird & Bird has asked me about Dr. Wright's statement that "*It is crucial to understand that nodes don't control the network; instead, they adhere to the inherent rules of the Bitcoin system. These nodes act as commercial agents tasked with enforcing the network's regulations"* (as is asserted in paragraph 15 of Dr Wright's first witness statement). For the reasons above, I do not agree with this statement.

{E/1/4}

Blockchain forks

76. More generally, Bitcoin participants may want to change various parameters of the system as it evolves over time. As a hypothetical example, consider a scenario in which the existing consensus among participants is that blocks should not take up more than 1 megabyte (MB), and suppose that some subset of participants decide that blocks instead should be smaller than 0.5 MB. The first type of miners (those who think blocks should be smaller than 1 MB) will accept as valid either type of block, since 0.5 MB is smaller than 1MB, whereas the second

set of miners will accept as valid only the smaller blocks (since 1 MB is bigger than their desired limit of 0.5 MB). In other words, this change is *backwards compatible*, as blocks following the new set of rules are still accepted by miners who are still following the old rules. More generally, any set of new rules that is backwards compatible in this way is called a *soft fork*.

77. A soft fork allows peers that are still following an older set of rules (because, for example, they have not yet upgraded their software to reflect the new set of rules) to participate in mining. Over time, however, even a soft fork could lead to a *chain fork*, as it would still be the case that newer miners would not consider the blocks produced by older miners as valid. In other words, it could happen that the network splits into two sets of miners: one following the old set of rules and one following the new set. This happens only if miners continue to produce blocks following both sets of rules. Thus, if a large enough majority supports the new set of rules, the rules will change according to this consensus and the old set of rules will be abandoned. This has happened at various points over Bitcoin's history.

78. We now consider a less hypothetical scenario in which the existing consensus among participants is that blocks should be smaller than 1 MB and some subset of participants decide that blocks instead should be smaller than 8 MB. The two requirements are not compatible, because blocks following the new set of rules may exceed the 1 MB block size, at which point they will be rejected by miners who are still following the old rules. Adopting a technical change of this kind will then result in two different blockchains which diverge at a single block. More generally, this type of backwards-incompatible change is called a *hard fork*.

79. This specific change (from a 1 MB block limit to an 8 MB block limit) led to a fork in the Bitcoin blockchain at block height 478,559 (which was mined in 2017) and the creation of an alternative cryptocurrency called Bitcoin Cash. Before this block height, the blockchains of Bitcoin and Bitcoin Cash are identical, but at this height and afterwards they are different, with one set of miners continuing to follow the consensus rules defined in Bitcoin and mining blocks with a size limit of 1 MB and the other following the consensus rules defined in Bitcoin Cash and mining blocks with a size limit of 8 MB.

Code forks

80. Peers participate in the Bitcoin network by running software, which means that changes to the consensus rules are implemented in this software and peers signal their support for a given set

of rules by choosing to run the version of the software that represents this set of rules. There are various pieces of software that can be used to participate in Bitcoin.

81. The most popular Bitcoin implementation, often called "Bitcoin Core,"[25] is maintained today in an open-source repository hosted at https://github.com/bitcoin/bitcoin. **Exhibit SM-22** {H/194} GitHub is a popular choice of version control system for the development of open-source software, and the code in this repository is also updated following a standard practice for open-source software:

    a. First, anyone can propose changes to the code by creating a *pull request*, which consists of their modifications to the various files that comprise the codebase and, typically, a short description of the changes and their motivation in making them. This pull request can be reviewed by anyone, in terms of providing comments, running tests to ensure the changes do not break the code or introduce any vulnerabilities, and assessing whether or not the changes seem likely to improve the software overall.

    b. Next, the pull request needs to be *approved* by a *maintainer* of the software repository. In general, software maintainers are developers who have demonstrated a willingness to be responsible for the repository in terms of providing timely reviews and thus enabling the code to update and continue to meet the needs of its users, and, as stated above, ensuring that any changes do not introduce vulnerabilities. In Bitcoin, as in many open-source projects, the set of maintainers has changed over time.

    c. Once a pull request is approved, it can be *merged* by the developer who created it or by a maintainer. At this point it is considered part of the Bitcoin Core codebase, but the changes won't be generally available to users of the Bitcoin Core software until a maintainer releases a new version of the software that incorporates all pull requests that were merged since the last version.

82. If someone wants to make a large change in Bitcoin, and in particular change the set of consensus rules (by, for example, using an alternative to proof-of-work or changing the block reward), they need to convince the broader community that this change is worthwhile and

[25] This name was first applied to the software in March 2014, upon its 0.9.0 release (see https://bitcoin.org/en/release/v0.9.0#how-to-upgrade). **Exhibit SM-23** {H/195}

ensure that it has sufficient support. One available process for this is to formally propose these changes as a *Bitcoin Improvement Proposal* (BIP), which are subject to more discussion from a broader set of participants than an average pull request due to the risk that making these changes would introduce a chain fork if they proved controversial. Like the Bitcoin Core codebase, BIPs are created and maintained in an open-source repository hosted at https://github.com/bitcoin/bips - **Exhibit SM-24**. BIPs were introduced in 2011 (see https://github.com/bitcoin/bips/blob/master/bip-0001.mediawiki **Exhibit SM-25**).

{H/196}
{H/197}

83. Suppose instead that someone wants to make a large change to the Bitcoin codebase (or, in general, the code of any open-source software released under an appropriate licence), in a way that is incompatible with its existing set of rules, and doesn't care about its existing users adopting these changes. They can still fork the code (i.e., duplicate it in a new repository), make the desired modifications, and then create software based on the modified code. This allows a developer to reuse any features they do like in the original software, rather than implement them from scratch, but have the freedom to change other things as they see fit. In the specific case of doing this with the code of Bitcoin Core or another Bitcoin client, creating and running this alternative software would have the effect of creating a new alternative cryptocurrency (again, assuming that the changes made were not compatible with the existing Bitcoin software and not adopted by its existing users). This is how many of the now thousands of alternative cryptocurrencies were created. Unlike cryptocurrencies like Bitcoin Cash that were produced as the result of a chain fork, this new cryptocurrency will not share any history with Bitcoin,[26] as its blockchain will be produced independently by peers who run the new software.

The Bitcoin P2P Network

84. In the paragraphs above, I have described how nodes in the Bitcoin network learn about transactions and blocks from peers in the network, and in turn inform peers about transactions and blocks that they have verified and believe to be valid. In this way, transactions and blocks can propagate throughout the peer-to-peer network and eventually be learned about by all peers in order to reach a consistent view of the blockchain. Further details of how Bitcoin works at the network layer are largely out of the scope of this report, but I discuss briefly here the question of how nodes discover new peers.

[26] The exception is if the new codebase keeps the same genesis block as Bitcoin.

85. When a user first downloads and runs Bitcoin software, their client is not connected to any other peers. Once they are connected to some peers, they can ask those peers who their peers are and attempt to connect to them (failing to do so only if those other nodes are offline or are already connected to a locally configured maximum number of peers).

86. In early Bitcoin implementations, clients connected to an IRC server (IRC is short for Internet Relay Chat and is to some extent the predecessor of modern instant messaging applications) and could both announce their own IP address and learn the IP addresses of other peers that they could then connect to. This approach does not scale well to a large number of users, however, so it was replaced in 2012. In this newer approach, clients are hardcoded with a list of URLs and make a request to a special type of server called a DNS server. This server will reply with a list of IP addresses, which the client can then attempt to connect to as peers.

Storing and using bitcoins

Local storage

87. In a typical usage, users store bitcoins in a *wallet*, which is a piece of software that stores private keys, keeps track of any associated UTXOs, and uses the private keys to form valid (i.e., validly cryptographically signed) transactions when users indicate that they would like to transfer their bitcoins to a particular recipient. This software is run on a computer or mobile device.

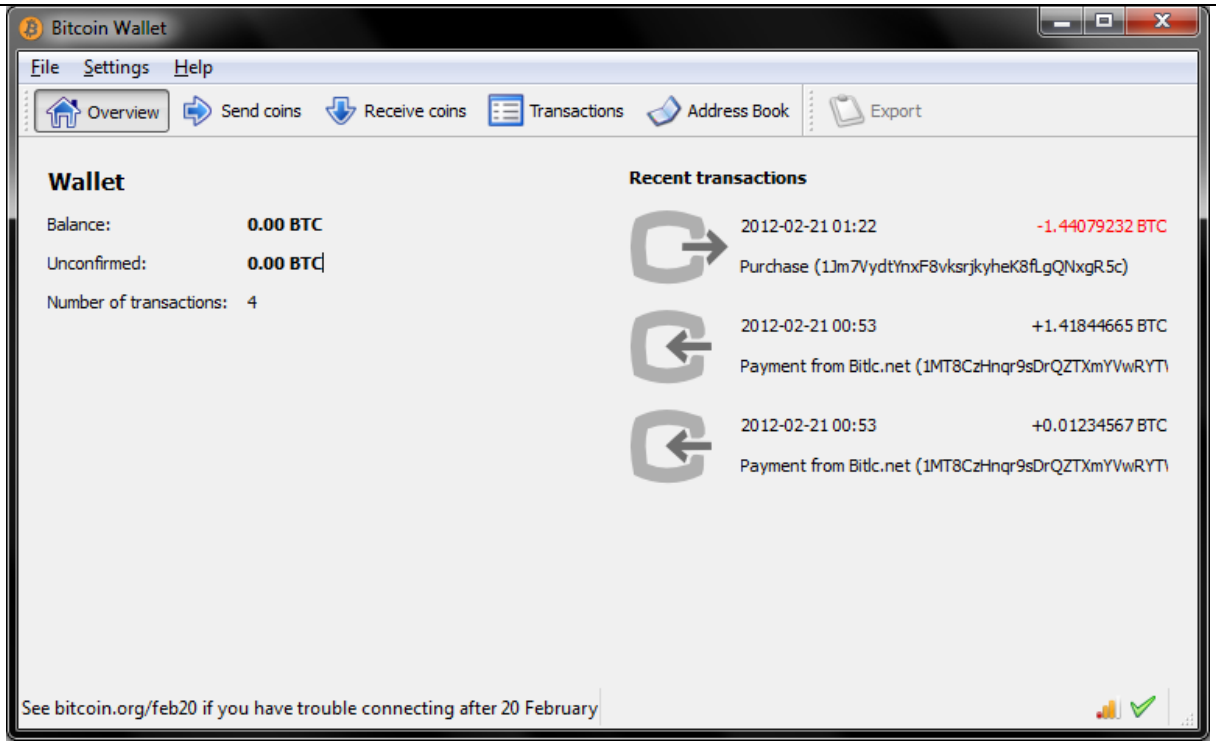


Figure 13. A screenshot of the Bitcoin Core wallet, taken from the Bitcoin Wikipedia page and available at https://en.wikipedia.org/wiki/Bitcoin#/media/File:Screenshot_of_Bitcoin-qt-0.5.2.png – ***Exhibit SM-26***

{H/198}

88. Figure 13 shows a screenshot of the Bitcoin Core wallet software (when it was known as Bitcoin-Qt). A user can see the transactions in which they have sent and received bitcoins using all the addresses maintained in this wallet; send bitcoins by specifying the address(es) of a recipient; and request bitcoins by generating a QR code that embeds the amount being requested and a specific address they control, which they can then share with the entity from whom they are requesting payment.

89. Losing access to the wallet means losing access to the bitcoins within it. Because of the severe consequences if a user loses their private keys, wallet software often provides a user with a *recovery phrase*. If the user keeps this phrase secure (either stored on a different device or written down using some physical medium), then even if the device they run the software on gets corrupted or erased, or they lose the device, they can still maintain access to their private keys by entering the recovery phrase into a freshly installed version of the software. While recovery phrases thus mitigate the risk of a user losing their bitcoins, there are still multiple risks associated with storing bitcoins on a personal computing device. For example, an attacker might be able to infect the device with malware; manipulate the user into giving them access to the device; or steal or gain access to the device and unlock it by guessing the user's password (or using their existing knowledge of it). Doing any of these

things could allow this attacker to access the wallet software and steal the user's bitcoins. I thus cover below four established ways of addressing this issue.

Custodial solutions

90. In traditional finance, many users choose to store their assets with a bank. In Bitcoin, many users make the same decision and store their bitcoins with *exchanges*[27] due to the convenience and the concerns around losing their bitcoins if they store them on a local device. On the other hand, users must trust the exchange to act as a custodian for their bitcoins.

Cold storage

91. To mitigate the risks associated with storing bitcoins on a device connected to the Internet and used for other purposes (and thus more susceptible to being infected with malware or otherwise compromised), users can store bitcoins in offline media, such as on a hardware wallet or just by writing private keys or recovery phrases on a piece of paper. This option is called *cold storage* and makes it almost impossible for these bitcoins to be stolen by cyberattacks (although the physical medium itself could still be stolen). On the other hand, it makes them difficult to transact with, as to transfer bitcoins stored in this way a user would have to first load the stored bitcoins into an online wallet. As such, the typical usage of cold wallets is in combination with "hot" / online wallets; for example, an exchange can protect any large reserves by keeping them in cold storage, while using a hot wallet to handle day-to-day transactions (i.e., deposits and withdrawals) with customers.

Multisignatures

92. In all the solutions described so far, there is a single point of failure: if the user loses their device or (for example) the piece of paper they wrote their private key on, or the exchange they store their bitcoins with goes down, then the user loses access to their bitcoins. To instead spread out trust, users can use a *multisignature* (multisig) address. Recall that a typical Bitcoin transaction accepts as valid only a signature produced using the private key associated with the input UTXO. For an m-of-n multisig input, there can be signatures required from any m out of n private keys. For example, in a 1-of-2 multisig, there are two participants, and as long as either of them produces a valid signature the transaction is

[27] These services are called exchanges because their primary purpose is to exchange Bitcoin and other cryptocurrencies for fiat currencies such as the pound or euro.

considered valid. This means that if a user stores one of the private keys on their laptop and the other on their mobile phone, then as long as they don't lose both devices they still have access to their bitcoins.

93. A multisig address typically appears in the Bitcoin blockchain starting with a 3, indicating that it is a P2SH (pay-to-script-hash) address. When spending it, the creator of the transaction must provide a redeem script of the form OP_PUSHNUM_M OP_PUSHBYTES_65 <pubKey1> … OP_PUSHBYTES_65 <pubKeyN> OP_PUSHNUM_N OP_CHECKMULTISIG, where the number M indicates how many signatures to expect that will verify against M of the N provided public keys. The transaction must also provide these M signatures.

Secret sharing

94. Multisig addresses have a different format from P2PKH addresses, and the set of participants is fixed at the time the address is formed. If a user in possession of a P2PKH address wants to share the ability to spend any UTXOs containing that address among participants, they thus have two options: first, they could do an on-chain transaction sending the bitcoins from their P2PKH address to a new multisig address. This has the downsides that the user would have to load their private key into an online wallet, and thus risk losing their bitcoins if the device the wallet is on Is compromised, and that the user would have to pay a transaction fee. Second, the user could split the private key for the P2PKH address using a cryptographic primitive called *secret sharing*. As with a multisignature, the private key can be split into shares so that any m of n participants can work together to reconstruct the original private key. The first and most common instantiation was given by the cryptographer Adi Shamir, so this is also often called Shamir secret sharing.

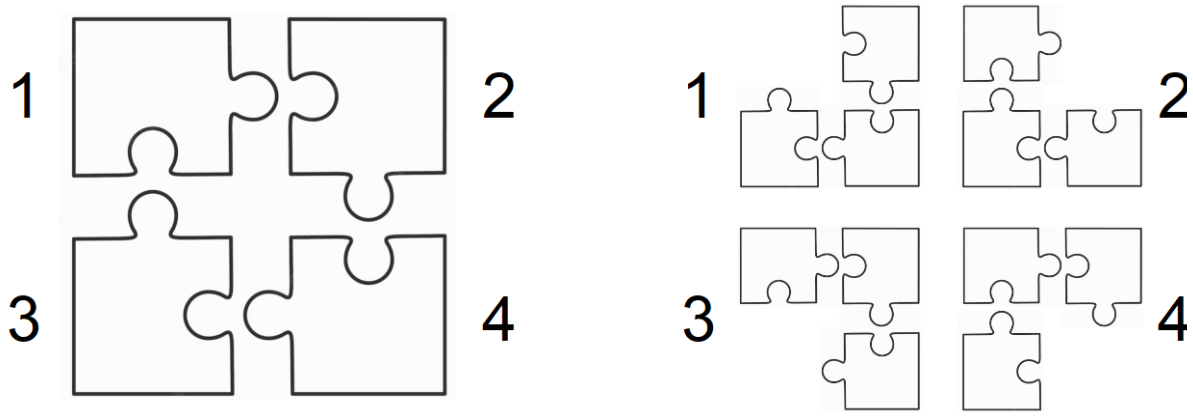


Figure 14. The two images above depict different ways to share pieces of a puzzle, which is analogous to a secret sharing scheme if we think of the secret as the completed puzzle. The image on the left depicts a 4-of-4 secret sharing scheme, as all four participants are needed to complete the puzzle. The image on the right depicts a 2-of-4 secret sharing, as any two of the participants can complete the puzzle on their own (i.e., without needing the pieces of the remaining two participants).

95. Figure 14 gives some intuition for how secret sharing schemes work. If we imagine the private key as a completed four-piece puzzle, we can think of giving out one piece to each of four people, as depicted on the left. In this case, the puzzle can be completed only if all four people use their pieces, making this a 4-of-4 secret sharing. On the right, we see an alternative way to give out pieces in which each person gets three pieces, with each different person missing a different piece. In this case, as long as any two of the people use their pieces, the puzzle can be finished, making this a 2-of-4 secret sharing.

Security of Digital Signatures

96. As described starting in the "Digital Signatures" section above, digital signatures are crucial for ensuring integrity in Bitcoin (by ensuring that users cannot spend coins they do not own) and in many other real-world applications. Given this, it is worth being clear about what security they do and don't provide.

97. Recall that a digital signature verification algorithm takes in three objects: a public key, a message, and a signature. Intuitively, it should be hard to produce a valid signature (i.e., one that passes verification) without knowing the private key corresponding to the public key.

However, the standard definition of security for digital signatures, unforgeability,[28] provides a weaker guarantee.

98. To understand why we provide only this weaker guarantee, consider the following scenario. A user **Alice** publishes a public key $\mathbf{pk_A}$, and is the only one who knows the corresponding private key $\mathbf{sk_A}$. Another user, **Bob**, asks Alice to sign the message "**Hi**", which she does: i.e., she gives him a signature **σ** such that Verify(pk_A, "Hi", σ) = 1.

99. Since the signature is just a series of bytes, there is nothing to stop Bob from giving it to any other user. For example, he could give it to a user **Carol**, who could check for herself that the signature is valid by running the verification algorithm. Thus, Bob was able to provide Carol with a valid signature under Alice's public key, but he did so by just copying the signature given to him by Alice rather than by knowing the private key himself.

100. This type of "*replay attack"* is not prevented by the digital signature itself. Instead, in practical scenarios where the goal is to use the signing process to establish possession of the private key, a user must be asked to sign a *new* message. In other words, asking a user to produce a signature on a message of their own choice is not meaningful because they may have just taken that signature from some public source or via direct interaction with the true owner of the private key.

Verification of digital signatures

101. Although the verification algorithm is known and standardised for any digital signature scheme, the algorithm is not applied in a vacuum and the circumstances in which it is used mean that there are additional requirements for its output to be trusted. Bird & Bird has asked me to explain the various requirements and levels of trust implicit in the steps of a signature verification process.

102. Beyond replay attacks or other cryptographic measures, there are many ways a seemingly valid signature could be produced under a given public key without actually knowing the corresponding private key. Most notably, if the user controls the software performing signature verification or if the software contains bugs, they can make it look like the signature has verified when in fact it hasn't. Trust in software can be increased in a variety of ways: audits, publishing the code in open source, trusting the reputation of known developers,

[28] More precisely, the definition of security is *(existential) unforgeability against a chosen message attack*, or (E)UF-CMA for short.

downloading the software from a known reputable source, etc. Notably, this is not a problem that can be solved cryptographically.

103. To use digital signatures as a way to establish the current possession of the private key corresponding to a public key by a given user, there are thus several requirements which must either be fulfilled, or must be replaced by assumptions or trusted sources.

    a. **Unique message.** The "challenger" (i.e., the entity wishing to establish possession) must ask the user to sign a message that they are sure has not been signed before for this public key, to prevent a replay attack. An example of a message that is unlikely to have been signed before is the current date and time. If the message that is signed is not one chosen by the challenger and known to be unique, then the requirement is not fulfilled and the process implicitly requires the challenger to *trust* that the message is unique.

    b. **Method of and result of verification.** The challenger must then run the verification algorithm using the public key, the new message, and the signature given to them by the user. If they do this process manually (i.e., by implementing the algorithm themselves on paper), then if the algorithm results in the signature being verified, it means that either the user is in possession of the private key or they found a way to forge signatures without access to the private key. This latter case is believed to be intractable (i.e., not practically possible) for all standardised digital signatures.

    c. **Semi-manual verification**. Verification is rarely, if ever, performed on paper, due to the size of the numbers involved. If instead the individual steps are performed using software tools on a computing device (such as using Microsoft Excel to do individual calculations, or implementing the verification function in some programming environment), then the user trusts that the computing device is accurately and faithfully performing each step and that their implementation accurately captures the abstract specification of the verification algorithm. Again, this is rarely done in practice due to the complexity of the algorithms involved (e.g., for ECDSA one would need to implement elliptic curve multiplication from scratch).

    d. **Software integrity.** More commonly, the challenger runs the verification algorithm using an existing implementation, in the form of software or a code library. In this case, the challenger must trust a range of factors:

i. They must trust that the implementation is running the correct verification algorithm before returning its output. If the implementation is buggy, it may unintentionally fail these requirements; if it was intentionally subverted, then the output cannot be trusted.

ii. They must also trust that the implementation is performing verification using the correct public key, message, and provided signature.

iii. If the challenger trusts a specific version of a piece of software, perhaps because they have inspected its source code, then they could have some faith that these trust assumptions held, especially if they download the open-source code from a trusted source and compile it into the software performing verification directly (and then pass it the expected input themselves). However, if the software is obtained from a remote source over the Internet then the challenger must trust (1) the source from which the software (or code) is being downloaded and (2) the data connection over which the download takes place. If the software is sourced from an untrustworthy place or downloaded over an insecure data connection, then the internal operation of the downloaded software could be different from what is expected.

e. **Software and hardware integrity of the computing device**. The challenger must also trust that the hardware, operating system, and software of any computing device that is used for verification is configured appropriately. For example, the challenger must trust that when the device is instructed to open the verification software, the version that is opened is the same one that was downloaded from the trusted source; or that the device displays the correct output that is passed to it by the verification software. Where a challenger uses a device in their own control, it is possible to control or verify its configuration appropriately. Where a device is not in the control of the challenger, these things must either be verified or assumed.

What a verified signature indicates

104. As discussed above, under the right circumstances it is possible to use the process of producing a digital signature to establish possession of the private key corresponding to a known public key. If the public key has some known associated owner, then producing a valid signature under this public key on a new message can be interpreted in one of the following ways.

a. The person producing the valid signature is the owner associated with the public key in question.

b. The person producing the valid signature was provided with a copy of the private key, being trusted by its original owner.

c. The person producing the valid signature has some kind of "oracle access" to the key's owner; i.e., when they get asked to sign a message they forward that request to the owner and use the signature they provide in response.

d. The person producing the valid signature obtained the private key in question by hacking, or compromised the security of ECDSA.

105. I am not aware of any suggestion of hacking or compromising ECDSA for any of the public keys associated with the creator(s) of Bitcoin, Satoshi Nakamoto, whose relevance I discuss below. I thus do not take them into account in what follows, although as discussed in Paragraph 20 it is possible to learn the private key from an ECDSA signature if the signature's randomness is generated in a non-uniform way.

The key in question

106. As described above, providing a signature that verifies with respect to a public key associated with some known owner constitutes evidence of some association with that owner (i.e., being the owner themselves or having access to them or the private key). It is thus worth considering which public keys are considered to be associated with Satoshi Nakamoto, the creator(s) of Bitcoin.

107. I understand that none of the keys used in the signing sessions were the PGP key believed to belong to Satoshi Nakamoto,[29] and that instead a public key from the blockchain was used. I therefore focus on blockchain-based signatures and public keys in the following.

[29] As far as I can determine, the PGP key used by Satoshi Nakamoto is the one at http://bitcointalk.org/Satoshi_Nakamoto.asc. **(Exhibit SM-27)** I made this determination by looking at the (identical) keys described as being associated with Satoshi at https://web.archive.org/web/20110228054007/http://www.bitcoin.org/Satoshi_Nakamoto.asc **(Exhibit SM-28)** (pointed to from https://bitcointalk.org/index.php?topic=458.msg5772#msg5772 on 25 July 2010 (**(Exhibit SM-29)**), and https://www.metzdowd.com/pipermail/cryptography/2015-November/027119.html. **(Exhibit SM-30)**

{H/199}
{H/200}
{H/201}
{H/202}

108. There is only one public key in the blockchain that could belong only to the creator(s) of Bitcoin, which is the public key used in the coin generation transaction of the genesis block (which I describe in "The Role of the Blockchain" section above). Any other block could technically have been produced by any other person who started Bitcoin mining early on. Thus, the most definitive evidence that one could provide to prove that they possessed a key associated with Satoshi Nakamoto would be to produce a new signature corresponding to the public key used in the genesis block's coin generation transaction, in the manner described in the "Verification of Digital Signatures" section above.

109. That public key has to date never spent its contents,[30] and in fact it cannot spend its contents because the software does not treat this transaction output as a UTXO. As such, it is not clear if anyone knows or has ever known the associated private key.

110. Another block that the Bitcoin community generally accepts as having been mined by Satoshi Nakamoto is the block at height 9, due to the fact that 10 bitcoins were later sent to Hal Finney from the public key associated with the output of the coin generation transaction in this block and that this transfer is generally believed to have been initiated by Satoshi Nakamoto. The fact that an outgoing transaction took place indicates that a private key is (or was at the point of the transaction) known. Therefore, a signature on a new message that verifies under the key used in the coin generation transaction for the block at height 9 would be indicative of association with Satoshi Nakamoto, at least within the general belief mentioned above. I refer below to the transaction transferring coins from Satoshi Nakamoto to Hal Finney as the SNHF Transaction and to the coin generation transaction for the block at height 9 as the Block 9 Generation Transaction.

The "Signing Sessions" with Journalists and with Gavin Andresen

111. Bird & Bird has also asked me to comment on what are known as the "signing sessions" between Dr Wright and Gavin Andresen, and between Dr Wright and various journalists. In this section, I do so based on the information that has been provided to me (specifically, the materials listed in the "Instructions" section of this report above).

[30] See https://btcscan.org/tx/4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b?expand, which shows that the contents are unspent (in the "spending tx" field) – **Exhibit SM-31** {H/203}

General comments on Dr Wright's technical explanations

{E/2} 112. Dr Wright's second witness statement begins by providing background on the technical terms and processes he describes. I therefore comment first on that technical background before discussing the signing sessions themselves. In this section, paragraph references refer to paragraphs in Dr Wright's second witness statement. I do not address all of Dr Wright's explanations here, but focus only on what is relevant to this section of my report.

{E/2/4} 113. I do not agree with Dr Wright that the signing and verification of messages is unique to Bitcoin, as stated in paragraph 4. Bitcoin uses ECDSA, which is a standardised pre-existing signature scheme, and there are other standardised signature schemes, such as RSA. There are many other examples of software and cryptographic libraries that implement signing and verification (for, again, both ECDSA and other signature schemes).

{E/2/4} 114. In paragraphs 7 to 11, Dr Wright describes use of the Bitcoin Core software to verify and sign a message. While it is correct that Bitcoin Core does include signing and verification functions, the Bitcoin Core software is not actually needed for the blocks mentioned in Dr Wright's statement. This is because the coin generation transactions for these blocks were P2PK transactions, meaning they contained the full public key (and not an address derived from the public key). This means all that would be needed to sign and verify a message with the keys in question would be an implementation of ECDSA (over the specific elliptic curve used in Bitcoin), which could operate directly on the keys themselves without the need to convert them into Bitcoin addresses. If Bitcoin Core is used, there needs to be an additional step to convert the public key to a Bitcoin address, but that is not mentioned in Dr Wright's statement.

115. Although Dr Wright emphasises the use of a Windows laptop (for signing) and a virtual machine running Linux (for verification), that setup is not required. To ensure the integrity of the signature verification process, it is important to ensure the integrity only of the setting in which verification takes place. This is because verification could be corrupted to falsely indicate success (regardless of whether or not the signing process was corrupted), whereas if verification is operating correctly then it is not possible for a signing process (again, corrupted or not) to pass verification without producing a valid signature (and a valid signature is not something that a corrupted signer can just happen to generate). In other words, using two

different environments doesn't add anything to the process, because all that matters is the integrity of the setting in which we perform verification.

{E/2/5} 116. In Paragraph 9, Dr Wright explains that it is crucial for the Bitcoin Core software to download the blockchain when it first starts. I do not agree that this is crucial for signing.

a. First, as I mentioned above, it is not necessary to use Bitcoin Core at all, as any software or code library implementing ECDSA could be used.

b. Downloading the blockchain is important if you want to participate in Bitcoin (such as verifying, receiving, and sending transactions), but it is not needed for signing. All that is needed for signing are the relevant keys or addresses and the message.

c. Signing can be carried out in Bitcoin Core before the blockchain has been downloaded. The download process can also take several days or weeks (due to the size of the blockchain), but during this time command line functions such as signing and verifying are available.

d. If the blockchain were downloaded, it would be possible to obtain public keys and addresses from the local copy of the blockchain as a first-party source. However, it is also possible to obtain them from third-party sources such as an online block explorer (of the kind that Dr Wright mentions in his second statement). In general this requires trusting the third-party source to provide the right keys or addresses, but for keys and addresses that are already known to the verifier (such as the "historical" keys used in the signing sessions) this risk is mitigated. Thus, the time-intensive process of downloading the whole blockchain is not strictly required.

e. I therefore do not agree that downloading the blockchain in the way described bolsters security of the signature verification process.

{E/2/5} 117. In paragraph 11, Dr Wright describes the output of the digital signature process as: "a message digest (message hash), referred to as a 'digital signature'." However, I do not agree with this definition:

a. A hash and a signature are different both procedurally and in what they were designed to achieve.

{E/2/10}
{E/2/10}

b. On a procedural level, as described in Paragraph 30, hashing means taking one input and producing one output in a deterministic manner. As described in Paragraph 32, a signing algorithm takes two inputs (a private key and the message to be signed) and produces one output (the signature). If the signing algorithm is randomised, as it is for ECDSA, the resulting signature is different every time the algorithm is run.

c. In terms of what they were designed to achieve, a hash is designed to succinctly represent an input of arbitrary length, while a signature is designed to bind a message to a public key, and thus to a specific party (the one in possession of the associated private key).

{E/2/8}

118. In paragraph 22, Dr Wright mentions that the use of the methods he describes limits the risk of exposing one's private key. I disagree that this is a particular feature of the methods he describes over any other methods for signing and verification: the purpose of a digital signature is to be given out freely (for example, you see multiple signatures from a website operator every time you visit a website secured using SSL / TLS), so there is no risk of exposing your private key by giving out signatures. Also, as I have explained above, the integrity of the process is also not bolstered by splitting the signing and verification processes across different systems, as the integrity of the process depends solely on the integrity of the setting in which verification is performed. Finally, the use of the term "decentralising" here does not match the ordinary usage of this term, particularly because the main feature of digital signatures is to allow verification by parties other than the signer; it is thus typical (but not required) that signing and verification happen on different computers.

{E/2/6}

119. In paragraph 12, Dr Wright describes the verification process as combining the "original message, message digest and the 'digital signature'." I do not agree that is an accurate explanation of the verification process.

a. It is correct that verification takes three inputs, but they are different to the inputs Dr Wright mentions. They are the message, the public key, and the digital signature itself.

{E/2/6}

b. In paragraph 12, Dr Wright refers to a "digital signature" and a "message digest" as if they are two different inputs, but in paragraph 11 he implies that the terms "message digest" and "digital signature" are used interchangeably.

My understanding of the journalist sessions and the Jon Matonis session

120. My understanding of how the signing sessions with the journalists and with Jon Matonis were conducted is as follows:

a. Dr Wright used his personal laptop.

b. He was using Windows, but he ran a virtual machine on the same laptop. The virtual machine was using a Linux operating system.

c. Both the Windows and the Linux systems shared the same storage partition (the 'C:' drive).

d. The virtual machine is referred to as a "hypervisor" virtual machine. A hypervisor is a process that manages the creation and operation of virtual machines. As such, it is unclear to me what this designation means, or what types of permissions were granted to the virtual machine. This information is not provided.

e. Bitcoin Core was downloaded, installed, and allowed to download the whole blockchain on both machines.

f. Dr Wright states that he signed a message of a speech by Jean-Paul Sartre which was stored in a file named "Sartre.txt", using the private key corresponding to the public key used in the coin generation transaction in block 9.

g. The signature was then verified on the Linux virtual machine, which output "true."[31]

121. I have already mentioned above some technical aspects of the general part of Dr Wright's statement with which I disagree, which are also relevant to these sessions. Beyond those, there are some specific aspects of these sessions that I comment on below.

{E/2/9} {E/2/9}

122. First, in paragraph 26, Dr Wright refers to signing "each said message with the text of a speech by Jean-Paul Sartre (the 'Sartre.txt')." In paragraphs 27 to 31, however, the process being described involves signing the message "Sartre.txt" (i.e., that specific string of 10 characters, rather than the contents of any file with that name or the text of a speech). This

[31] The output "true" or "false" refers to whether or not the signature passes the verification test, rather than the actual message displayed on screen (which can vary depending on the implementation). For example, OpenSSL on Linux outputs "Verified OK" if the verification returns true and "Error verifying data" if the verification returns false.

does not affect the integrity of the process, as long as the same message was used consistently across signing and verification, so I offer this solely as a point of clarification.

{E/2/9} 123. In paragraphs 27 and 30, Dr Wright states that the commands he used were

```
bitcoin-cli signmessage "address_of_block_9" "Sartre.txt"
```

and

```
bitcoin-cli verifymessage "address_of_block_9" "received_signature"
"Sartre.txt".
```

124. I describe these commands in more detail below, in terms of what it means to run them (or see them run) on a computer.

a. The term "bitcoin-cli" tells the system to use the program that has been assigned that name. That name is assigned to the Bitcoin command-line interface (CLI) tools upon its installation, but this is an *environment variable,* which is a short string that is used to access a program by the computer's environment (i.e., the operating system running on the computer). It can be configured to point to any program, and it is possible for a user to change the program to which this variable points by typing a simple command.[32]

b. The terms "signmessage" and "verifymessage" indicate to the program the name of the function to be performed. In Bitcoin Core, these functions are expected to sign or verify a message (respectively). However, if the program to which the "bitcoin-cli" variable points is different, it is not possible to state what process is carried out when those function names are provided.

c. For example, it would be simple to develop a small program that outputs a random string, matching the pattern expected for an encoded signature (e.g., of the right length), when given the "signmessage" instruction, regardless of the provided inputs (e.g., the address and message).

[32] For example, in Linux operating systems including CentOS, the program to which the environment variable "bitcoin-cli" pointed could be changed by entering "`export bitcoin-cli=[/path/to/program]`" into the command line, where [/path/to/program] is the path to the program to be used (which could be any program, for example one created by the user). Alternatively, instead of entering the command, it could be stored in the user's profile (which is saved as an editable text file). For an example of how an environment variable can be assigned to a custom script, see paragraph 138 below which refers to Dr Wright's verification script.

d. It would also be simple for the same program or a different program to simply output "true" whenever given the "verifymessage" instruction, again regardless of the inputs. As described above, it would also be simple to change the environment to have the "bitcoin-cli" variable point to this program.

e. As I understand it, there was no independent check of the environment variables or integrity of the program used during these sessions.

Security with respect to the private key(s)

{E/2/17} 125. In paragraph 61 of his second statement, Dr Wright discusses the status checks, backups, and maintenance carried out on his version of Bitcoin Core. I have the following comments on this paragraph:

a. The paragraph describes various operations associated with the maintenance of the Bitcoin Core software. Because it is in the section referring to the installation of Bitcoin Core on the Eurocom laptop used in the signing sessions (at paragraph 42), I assume the described operations were performed on the same laptop.

{E/2/13}

b. The paragraph describes ensuring that Bitcoin Core "remained connected to the network and updated with the most recent blocks" (61a) and mentions "scheduled checks for software updates" (61b). This suggests that the laptop was connected to the Internet, in either a persistent or scheduled way.

{E/2/17}
{E/2/18}

c. From a security perspective, it is surprising that Dr Wright would consistently connect a system that stores such sensitive private keys to the Internet. This is especially true given the wide availability of "cold storage" solutions (described in the "Cold storage" section above) and the complex measures that he describes having taken to divide the key into secret shares (in paragraphs 140-143 of his first witness statement).

{E/1/26}

d. Likewise, Dr Wright's reference to keeping "periodic backups of the Bitcoin Core wallet and data" indicates that these sensitive private keys would be stored in these backups in addition to the laptop, which is another source of risk. By contrast, the risk of exposing a private key by providing someone with a signature is far lower (effectively zero, as described above).

126. In my opinion, connecting a computer to the Internet with such sensitive data residing on it in this way does not appear to be consistent with the other security measures that Dr Wright describes elsewhere in relation to the signing sessions.

My understanding of the Gavin Andresen session

127. My understanding of the "signing session" carried out with Gavin Andresen is as follows. This understanding is based on the materials I have read (explained in the Instructions section of this report), and Bird & Bird has prepared this list below following our discussion of those materials. In several cases the accounts differ, as indicated in the list below.

a. **Location**: Gavin Andresen travelled to London to meet Dr Wright for the session. The meeting took place in a hotel in London, which is named as a "Covent Garden hotel, in a room downstairs" in ID_004572. {ID_004572}

b. **Notes:** There are no notes from the session itself, but Gavin Andresen stated that the best notes he has are in the form of a written discussion on Reddit (which is ID_004572). {ID_004572}

c. **The overall process** involved Dr Wright signing a message on his laptop, transferring the signature to a new laptop, and verifying the signature on that laptop.

d. **Hardware used**: Dr Wright used his personal laptop to produce the signature. Gavin Andresen states in his deposition that he had brought his own laptop to verify the signature, but that was not used. Instead, an assistant went out and returned with a laptop, which seemed new. I refer to this as the "*Signing Session Laptop*" below. Gavin Andresen did not accompany the assistant or check that it was factory sealed.

e. **Internet connection used:** It is not clear what Internet connection was used. Dr Wright states in his second statement that the hotel WiFi was used. However, Gavin Andresen stated in ID_004572 that "it's very possible we used a hotspot because the hotel WiFi was flaky down in the basement." {ID_004572}

f. **Installing Windows on the Signing Session Laptop**: It is not clear who set up the Signing Session Laptop. In ID_004572, Gavin Andresen indicated that the laptop {ID_004572}

came with Windows pre-installed and wrote that "I think I remember Craig being please it wasn't Win10 (but disappointed they couldn't get a Linux on short notice)." However, Dr Wright states that "Gavin took the lead in setting it up from scratch. This necessitated an operating system installation" (paragraph 36 of his second witness statement).

{E/2/11}

g. **Installing software on the Signing Session Laptop:** In his deposition, Gavin Andresen states that "Craig downloaded and installed software." However, Dr Wright states in his first witness statement that "Gavin […] installed Electrum" and in his second statement that "He began by downloading Electrum."

h. **Choice of software:** The Electrum wallet software was used for signing and verification. Gavin Andresen states in his deposition that Dr Wright chose to use Electrum. However, Dr Wright states that "Gavin opted for Electrum."

i. **Source of Electrum software:** It is not clear what source was used to obtain the Electrum software. In ID_004572, Gavin Andresen stated that they initially attempted to download and compile Electrum from its source code, but that failed and then they moved on to download the software binary (i.e., pre-compiled software ready to be run by the computer). However, Dr Wright's second witness statement suggests that he began by downloading the binary: "Gavin followed standard software installation best practices. He began by downloading Electrum directly from the official website. Before proceeding with the installation, he verified the integrity of the downloaded software by comparing its hash value with the one provided on the website." Dr Wright also states that he cannot be certain of the exact version of Electrum that was installed.

{ID_004572}

j. **Integrity of Electrum download:** It is not clear whether or not the integrity of the binary was verified at the point of download. Dr Wright states that it was, using the hash value on the website. However, Gavin Andresen was asked whether he verified the hash digest of the download against something he brought with him independently, and answered "No, I did not" (which does not mean that the download could not have been verified against the website hash as stated by Dr Wright).

k. **Choice of message:** It is not clear who chose the message to be signed. Gavin Andresen stated in his deposition that he chose part of the message but that Dr Wright

added the letters "CSW" at the end of it. Dr Wright states that the original message was "chosen by Gavin."

l. **Source of public key:** Keys from early blocks (i.e., blocks at low heights) in the Bitcoin blockchain were used for signing. It is not entirely clear, however, which of them were used. In Dr Wright's first and second statements, he refers to using the keys from multiple blocks, but Gavin Andresen appears to refer to signing using only one public key (the one in the Block 9 Generation Transaction).

m. **Transfer of the signature between laptops**: After signing the message on Dr Wright's laptop, the resulting signature was transferred from Dr Wright's laptop to the Signing Sessions Laptop. It is not clear how this took place. In his evidence in Norway, Dr Wright stated that the keys were read out verbally and typed in. However, in his first and second witness statements, Dr Wright stated that "Gavin used his USB drive to transfer the signatures" and that "I copied and pasted this 'digital signature' to Windows notepad.exe and saved the file on a USB key…and gave it to Gavin who inserted it into his new laptop."

n. **Source of public keys.** The public keys used to sign were sourced from the Bitcoin blockchain, but it is not clear how those keys were loaded onto the laptops that were used, which is not mentioned in the accounts I have seen. Dr Wright's witness statement suggests that they were taken from an online block explorer. Gavin Andresen mentions in his deposition that he brought with him a list of all the public keys from early blocks and used that to verify the keys.

o. **Verification.** On the Signing Session Laptop, Dr Wright opened a piece of software (apparently the software that had been downloaded as mentioned above) and fed into it the data from the USB drive. Dr Wright then manually typed in a message. There was an error the first time he did so, so he then typed in a message a second time. The software displayed output information (apparently a confirmation that the verification process had been successful), which led Mr Andresen to believe that the data on the USB drive represented a valid signature of the message in question, signed under the private key corresponding to the public key used in the Block 9 Generation Transaction.

Possibility of subversion of the Gavin Andresen session

128. Whether or not the process of verifying a signature can be trusted depends on whether the process is likely to have been subverted in some way. I therefore first consider whether the verification process was vulnerable to being subverted, and to what extent it would be possible for a technically skilled person to perform any potential subversions. It is helpful to revisit the ways in which the process of verifying a digital signature can be subverted, as discussed in "The Security of Digital Signatures" section above.

129. **Replay attack.** Given that the message being signed in the session was (according to my understanding) "*Gavin's favorite number is 11-CSW*", it appears to contain information personal to Mr Andresen and not of a kind that I would expect to have been used before. I therefore do not consider it likely that it was subject to a replay attack as described above.

130. **Software integrity.** In terms of the software, however, there are a number of ways in which it would have been possible to use software that did not in fact perform any signature verification but nevertheless made it look like it had been presented with a valid signature. These include the following.

a. The software that was downloaded and run could have been something other than the intended Electrum wallet software. This could have happened by downloading it from somewhere other than the genuine Electrum website or GitHub repository. For a skilled software developer, it would be easy to create software that looked identical to the genuine Electrum software but had slight differences in its functionality, especially given that Electrum is open-source software (i.e., its source code is publicly available).

    i. It would have been possible to guarantee the integrity of the website by checking various details (for example, by inspecting its HTTPS certificate and URL carefully).
    ii. Likewise, it would have been possible to guarantee the integrity of the software by checking that a hash of the downloaded software was equal to a known hash of that version of the Electrum software, as obtained from a reliable source.
    iii. It is my understanding that none of these checks were performed.
    iv. Dr Wright does mention that a hash of the file was checked but that this was from the same source that the software was downloaded. This check is a good way to ensure that software has not been corrupted in the process of being downloaded. However, it is not a good way to ensure that the software is the one expected in some external sense, because the source of the hash and the source of the software are the same.

v. I have been provided by Bird & Bird with an archived copy of the official Electrum website from early 2016, at the URL https://web.archive.org/web/20160502203936/https://electrum.org/#home **(Exhibit SM-32)**. Here, the "download" page includes links to .asc files containing PGP signatures. However, Dr Wright stated in his evidence in the Norway proceedings that Mr Andresen did not "do the signature file using PGP, because that would require using PGP" but that he did get "the check-sum" (meaning, the hash of the software binary). There are no links provided on the official Electrum website beyond the .asc files, which means that either (1) the software was downloaded from somewhere other than the official Electrum website or (2) the checksum / hash was obtained by manually decoding the PGP signature and extracting the underlying hash.

b. The downloaded software could have been the real Electrum software, but other software (such as malware) could have been running on the laptop that interfered with its expected behaviour. This could have been prevented by ensuring that the Signing Session Laptop was factory sealed and monitoring the setup process, but it is again my understanding that these checks were not performed. It is therefore possible for malware to have been installed on the laptop in a number of ways:

i. First, since the Signing Session Laptop was not verified as being factory sealed, additional software could have been installed prior to being brought into the room.

ii. Second, since my understanding is that it took hours to set up the new laptop, malware or other software could have been downloaded and installed during this setup process.

iii. Third, additional software could have been transferred to the Signing Session Laptop via the USB drive that was plugged into it.

iv. Fourth, it is possible that software could have been introduced through the network connection, as I explain below.

h. The integrity of the network connection is another way that the download of the Electrum software could have been compromised, or that malware could have been introduced onto the Signing Session Laptop.

    a. For a software download process to be trusted, it is implicitly necessary to trust the network connection that is used for that download.

    b. The Signing Session Laptop could have connected to the public hotel WiFi network (as is suggested by Dr Wright). In this case it would be necessary to trust the integrity of the network.

    c. The Signing Session Laptop could have connected to a network designed to look like the public hotel WiFi network but that was in fact set up by someone wishing to compromise the Signing Session Laptop. In this case it can be assumed that the network does not have any integrity (i.e., that it is compromised).

    d. The Signing Session Laptop could have connected to a hotspot (as is suggested by Mr Andresen). In this case it would be necessary to trust that the device used to provide the hotspot was configured in an expected way.

    e. In any of these cases, if the network were compromised, it would be possible for a remote attacker to install malware or other software on the Signing Session Laptop in a variety of ways: for example, by serving malware instead of the Electrum software, or by having the user consent to its download in a pop-up window designed to look like the captive portal commonly used when connecting to public WiFi networks. Carrying out such an attack would require more domain knowledge in security than the attacks described previously, but is still entirely feasible.

    f. I do not understand there was any reason for the network to be deemed as trusted.

Overall opinion on the signing sessions

131. In my view, the evidence provided in the signing sessions cannot be considered as reliable in establishing possession of the private key(s) corresponding to the public key(s) used.

132. I first consider the signing sessions with the journalists and with Jon Matonis. As described in the "My understanding of the journalist sessions and the Jon Matonis session" section above, it would be simple for someone with any degree of technical expertise to write a new "verifymessage" function that output "true" given any inputs, and to then change the environment to point the "bitcoin-cli" variable at the program containing this function (and to do something similar on the signing side). My understanding is that no checks were carried out to ensure that this did not happen, which means these demonstrations did not provide any additional technical evidence of Dr Wright's possession of the relevant private key given the ease with which they could be subverted.

133. In terms of the signing session with Gavin Andresen, I reach a similar conclusion in light of the limitations described in the "Possibility of subversion of the Gavin Andresen session" above: while subversion would have been more technically difficult in this case, there were many ways in which trust was assumed rather than verified. Namely:

    a. It would be technically straightforward to create software designed to look like Electrum that would provide a misleading output upon running signature verification. The integrity of the downloaded software could have been verified but doesn't seem to have been, so it would also be straightforward to install this software on the Signing Session Laptop instead of the genuine Electrum software.

    b. It would be technically feasible to create a program that would interfere with the operation of other software running on the operating system, and more specifically that would interfere with the (genuine) Electrum software and cause it to provide a misleading output. The integrity of the Signing Session Laptop could have been verified but doesn't seem to have been, so it would have been straightforward to run this program on the Signing Session Laptop and thus cause the Electrum software to provide a misleading output.

    c. Furthermore, even if there had been some integrity checks conducted on this laptop, it still would have been possible to install this type of program on the Signing Session Laptop due to the fact that there were no integrity checks performed on the accessed Internet connection.

d. Ultimately, many of these avenues of risk could have been ruled out but were not, and thus there is a meaningful possibility that Dr Wright did not produce a valid signature during this interaction.

The "Sartre Blog Post"

134. Bird & Bird has also referred me to a blog post created by Dr Wright, which I refer to as "the Sartre Blog Post," and asked me to comment on the information set out there. I have also been asked to consider whether there are any relevant technical differences between this and the signing sessions that cause me to reconsider any part of my opinion about the signing sessions. My comments and opinions are as follows.

Replayed data in the Sartre blog post

135. Before I describe the components of the blog post, there are two encoding schemes that it is useful to be aware of. Some of these have already been used throughout my report.

a. The **hex** (or, base16) encoding scheme converts binary digits to sixteen different symbols. These are typically the symbols '0' to '9' and 'A' to 'F'.

b. The **base64** encoding scheme converts binary digits to, as the name suggests, 64 different symbols. These are typically the symbols '0' to '9', 'a' to 'z', 'A' to 'Z', '+', and '/' (with '=' used for padding the string to the appropriate length).

136. I will go through the cryptographic components of the blog post in turn. I have been given multiple versions of the blog post, but these cryptographic parts are the same across all versions.

a. The string IFdyaWdod….GkuCgo=. This is the base64 encoding of the string " Wright, it is not the same as if I sign Craig Wright, Satoshi." (including the leading space character).

b. The string 479f9dff…68b05. As shown in https://gist.github.com/ryancdotorg/893815f426f181d838c1b44aa187f05a **(Exhibit SM-33)**, this is the modified transaction hash for Bitcoin transaction 828e…f509fe (namely, it is the hash of the part of the transaction that actually gets signed). This is the SNHF Transaction that is referenced in "The key in question" section above; i.e., it is the transaction that spends the

{H/205}

contents of a UTXO associated with the public key used as output in the Block 9 Generation Transaction. Thus, this hash can be derived deterministically from this transaction, and again is the part of the transaction that actually gets signed.

c. The string MEUCIQDBK…hcU+fM4=. This is a base64 encoded string that, when decoded and then encoded using hex, yields a string 30450221…5c53e7cce. This is the same as the string used as the signature in the same transaction referenced above, as is visible in the "JSON" view at https://www.blockchain.com/explorer/transactions/btc/828ef3b079f9c23829c56fe86e85b4a69d9e06e5b54ea597eef5fb3ffef509fe (**Exhibit SM-34**). In other words, this string is just a different representation of the signature used in the SNHF Transaction, (i.e. it is the same information written in base-16 instead of base-64).

d. The string 0411db9…6b412a3. As we can see at the detailed view at https://btcscan.org/tx/0437cd7f8525ceed2324359c2d0ba26006d92d856a9c20fa0241106ee5a597c9?output:0 (**Exhibit SM-35**), this is identical to the public key used in the coin generation transaction for the block at height 9.

137. To summarise, the blog post contains four main cryptographic objects. The first of these, the base64 encoding of a string, is not connected to the others and could be produced by anyone. The second of these acts as the encoding of a message **m**, the third of these as the encoding of a signature **σ**, and the fourth of these as an encoding of a public key **pk**. It is the case that Verify (**pk**, **σ**, **m**) = 1; i.e., that **σ** is a valid signature on the message **m** under the public key **pk**. However, as described above, all of these objects can be derived directly from the data for the SNHF Transaction and Block 9 Generation Transaction, which due to the nature of the blockchain are available to everyone. This data is thus replayed from these transactions, which as described in the "Security of Digital Signatures" section above means it provides no cryptographic evidence of the possession of the associated private key.

Script with additional variable in the Sartre blog post

138. Finally, I note that Dr Wright refers in the blog post to a shell script called "EcDSA.verify", which he states was the script used to verify the signature, shown below. This appears to run using an environment variable to assign a new "verify" command, in a similar way to that which I described above.

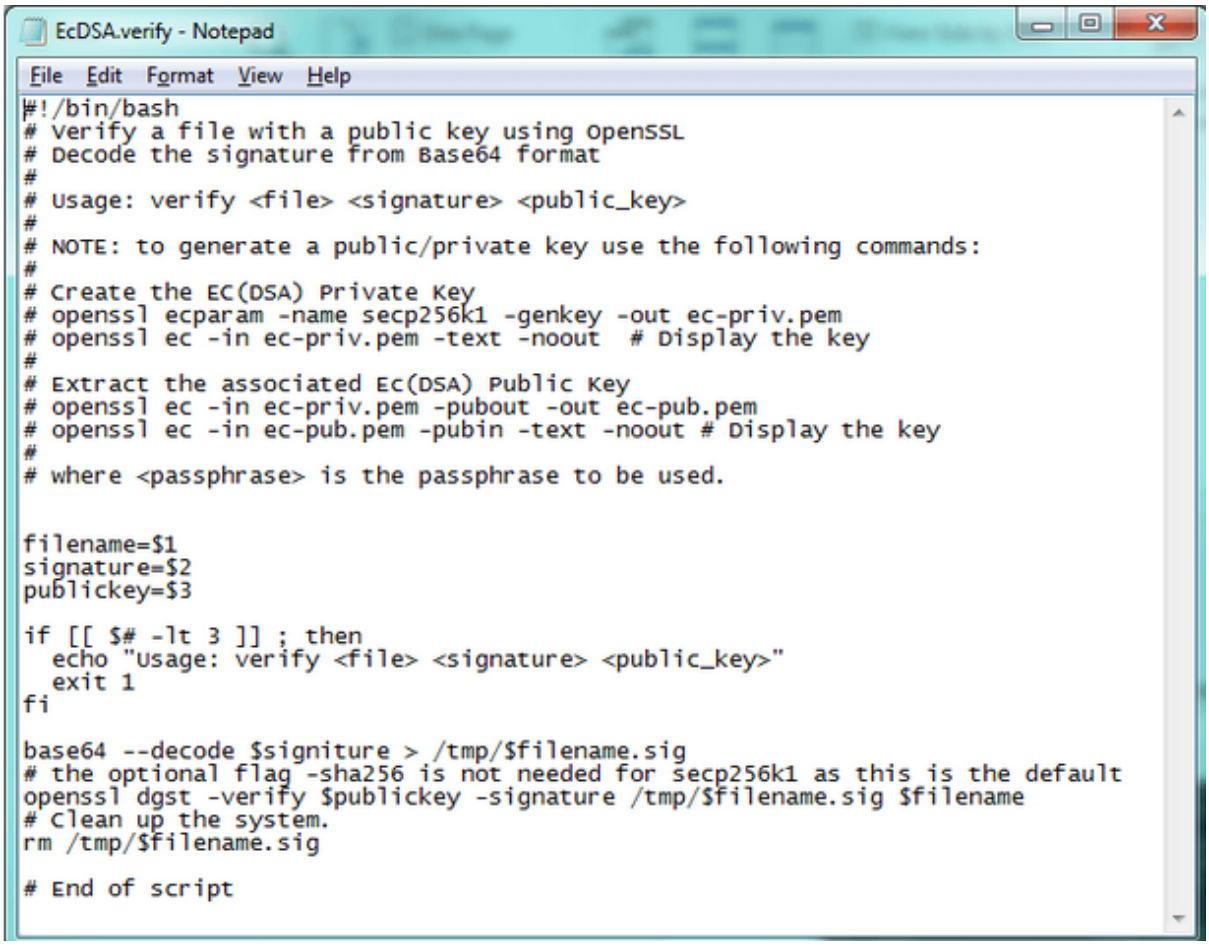


Figure 15: Script image from Dr. Wright's blog post

139. This script interprets three inputs passed to it and assigns the second one to a "`signature`" variable. However, when using the "`base64 --decode`" command, the input variable is spelled as "`signiture`". If the "`signiture`" variable had not been assigned to anything prior to the script being run, the script would not run properly. If instead the "`signiture`" variable had been assigned to an object, the script would run, but would perform verification using that object, instead of the signature passed to it as input (and the signature input would go unused).

140. This would allow someone to verify against a signature unseen to the (human) verifier, namely the one stored in the "`signiture`" variable. If verification passed for the "`signiture`" signature and a new message and public key, this would still act as evidence of the possession of the associated private key. If the message were not new, however, this would allow for a replay attack to be performed using the hidden "`signiture`" signature, despite appearing to the viewer as though a different signature were being verified within the script.

Declaration of Independence

1. I understand that my duty is to help the Court to achieve the overriding objective by giving independent assistance by way of objective, unbiased opinion on matters within my expertise, both in preparing reports and giving oral evidence. I understand that this duty overrides any obligation to the party by whom I am engaged or the person who has paid or is liable to pay me. I confirm that I have complied with and will continue to comply with that duty.
2. I confirm that I have not entered into any arrangement where the amount or payment of my fees is in any way dependent on the outcome of the case.
3. I know of no conflict of interest of any kind, other than any which I have disclosed in my report.
4. I do not consider that any interest which I have disclosed affects my suitability as an expert witness on any issues on which I have given evidence.
5. I will advise the party by whom I am instructed if, between the date of my report and the trial, there is any change in circumstances which affect my answers to points 3 and 4 above.
6. I have shown the sources of all information I have used.
7. I have exercised reasonable care and skill in order to be accurate and complete in preparing this report.
8. I have endeavoured to include in my report those matters, of which I have knowledge or of which I have been made aware, that might adversely affect the validity of my opinion. I have clearly stated any qualifications to my opinion.
9. I have not, without forming an independent view, included or excluded anything which has been suggested to me by others including my instructing lawyers.
10. I will notify those instructing me immediately and confirm in writing if for any reason my existing report requires any correction or qualification.
11. I understand that:
    a. my report will form the evidence to be given under oath or affirmation;
    b. the court may at any stage direct a discussion to take place between experts;
    c. the court may direct that, following a discussion between the experts, a statement should be prepared showing those issues which are agreed and those issues which are not agreed, together with the reasons;
    d. I may be required to attend Court to be cross-examined on my report by a cross-examiner assisted by an expert; and
    e. I am likely to be the subject of public adverse criticism by the judge if the Court concludes that I have not taken reasonable care in trying to meet the standards set out above.
12. I have read Part 35 of the Civil Procedure Rules and I have complied with its requirements. I am aware of the requirements of Practice Direction 35 and the Guidance for the Instruction of Experts in Civil Claims 2014.
13. I confirm that I have acted in accordance with the Code of Practice for Experts.
14. I confirm that I have made clear which facts and matters referred to in this report are within my own knowledge and which are not. Those that are within my own knowledge I confirm to be true. The opinions I have expressed represent my true and complete professional opinions on the matters to which they refer.

Signed: Sarah Meiklejohn

Date: 23 October 2023