Made on behalf of Defendant in COPA Claim
Made on behalf of Claimants in the Coinbase Claim, the Kraken Claim and the BTC Core Claim
Second Witness Statement Dr Craig Steven Wright
Dated 9 October 2023

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

Claim No. IL-2021-000019
(the "**COPA Claim**")

BETWEEN:

CRYPTO OPEN PATENT ALLIANCE

Claimant

- and -

DR CRAIG STEVEN WRIGHT

Defendant

Claim No. IL-2022-000035
(the "**Coinbase Claim**")

BETWEEN:

(1) DR CRAIG STEVEN WRIGHT
(2) WRIGHT INTERNATIONAL INVESTMENTS LIMITED

Claimants

- and -

(1) COINBASE GLOBAL, INC.
(2) CB PAYMENTS, LTD
(3) COINBASE EUROPE LIMITED
(4) COINBASE, INC.

Defendants

Claim No. IL-2022-000036
(the "**Kraken Claim**")

BETWEEN:

(1) DR CRAIG STEVEN WRIGHT
(2) WRIGHT INTERNATIONAL INVESTMENTS LIMITED

[illegible]

Claimants

- and -

(1) PAYWARD, INC.
(2) PAYWARD LTD.
(3) PAYWARD VENTURES, INC

Defendants

Claim No. IL-2022-000069
(the "**BTC Core Claim**")

BETWEEN:

(1) DR CRAIG STEVEN WRIGHT
(2) WRIGHT INTERNATIONAL INVESTMENTS LIMITED
(3) WRIGHT INTERNATIONAL INVESTMENTS UK LIMITED

Claimants

- and –

(1) BTC CORE

[redacted]

(16) BLOCK, INC.
(17) SPIRAL BTC, INC.
(18) SQUAREUP EUROPE LTD
(19) BLOCKSTREAM CORPORATION INC.
(20) CHAINCODE LABS, INC

[illegible]

(21) COINBASE GLOBA INC.
(22) CB PAYMENTS, LTD
(23) COINBASE EUROPE LIMITED
(24) COINBASE INC.
(25) CRYPTO OPEN PATENT ALLIANCE
(26) SQUAREUP INTERNATIONAL LIMITED

Defendants

SECOND WITNESS STATEMENT OF DR CRAIG STEVEN WRIGHT

I, CRAIG STEVEN WRIGHT, of [redacted] state as follows:

Introduction

1. I refer to my first witness statement dated 28 July 2023. I make this witness statement in response to requests 19 to 23 of the Combined Request for Further Information dated 23 June 2023. I make this statement on my own behalf and am also authorised to make it on behalf of the other Claimants in the Coinbase Claim, the Kraken Claim and the BTC Core Claim.

2. To prepare this witness statement, I was sent requests 19 to 23 and asked to consider them. I then prepared a draft statement. I discussed this with my solicitors, who then amended the draft to reflect my further comments and sent it to me for review and amendments. This process was then repeated until the witness statement was complete.

3. I have been asked to confirm the following:

   "The purpose of this witness statement is to set out matters of fact of which I have personal knowledge. I understand that it is not my function to argue the case, either generally or on particular points, or to take the court through the documents in the case. This witness statement sets out only my personal knowledge and recollection, in my own words. On points that I understand to be important in the case, I have

[illegible]

stated honestly (a) how well I recall matters and (b) whether my memory has been refreshed by considering documents, if so how and when. I have not been asked or encouraged by anyone to include in this statement anything that is not my own account, to the best of my ability and recollection, of events I witnessed or matters of which I have personal knowledge."

I confirm this is correct.

Background

4. Bitcoin's construction and infrastructure allows for a unique feature where messages can be 'signed' using a private key and then verified by others to ascertain the authenticity of the message origin. This can be instrumental in proving the possession and control of a Bitcoin address without revealing the associated private key.

5. In the demonstrations to Rory-Cellan Jones (BBC) and Ludwig Siegele (the Economist) and in the demonstrations to Jon Matonis and Gavin Andresen, I used this feature to demonstrate that I was in possession of the private keys associated with the early blocks using one of the two methods below:

   a. Bitcoin message verification (cross-system);
   b. Electrum message verification (across two computers).

6. I explain each method in general terms below.

Bitcoin Message Verification (Cross-System)

7. The preliminary action entails procuring the Bitcoin Core software. Where one device is configured to run using two different operating systems (for example Windows and a Linux environment operating on a hypervisor virtual machine), a version of the Bitcoin Core software compatible to both systems should be downloaded. The software should

[illegible]

be sourced directly from the official Bitcoin Core website to ensure the utmost security and authenticity. Avoiding third-party sources significantly mitigates the risk of encountering spurious or tampered versions.

8. Following this stringent sourcing criterion, the software should be downloaded and securely housed. For a device using two different operating systems, this should be housed in the shared partition, denoted as 'C:'. This systematic approach to acquiring the software underscores the dedication to procedural integrity and safeguarding system resources. Note that a device using Windows and Linux requires two downloads as the Windows and Linux software uses separate install files.

9. Once the Bitcoin Core software is successfully installed, it should be initiated to establish a connection to the Bitcoin network. The software begins the process of synchronising with the network, downloading and verifying the entire blockchain history. Given the size of the blockchain, this synchronisation process can be time-intensive. It's crucial for ensuring that the local instance of the software is up-to-date with all transactions on the network, bolstering security and functionality. Upon completion of this synchronisation, the software indicates its up-to-date status, affirming that it has the most recent record of transactions and blocks from the Bitcoin network.

<u>Message 'Signing' on the Origin System</u>

10. On the source system, using the Bitcoin Core command-line interface, one can create a 'digital signature' for a given message when that person is also the owner of the private key, as 'signing' is a function of identity and not mere possession:

```
bitcoin-cli signmessage "_bitcoin_address" "chosen_message"
```

11. Upon execution, the message is processed using the 'digital signature' algorithm. This command yields a message digest (message hash), referred to as a 'digital signature', which serves as a cryptographic proof that the message was 'signed' by the private key corresponding to '_bitcoin_address'.

[illegible]

Transferring the 'Signature' and Message

12. When combined with the original message, message digest and the 'digital signature', the output can then be transferred to any other system using conventional methods (e.g. by email, file transfer or even manual transcription). In the case of the demonstrations I refer to below I did this by copying between the Windows system and a Linux virtual machine. The essential element is ensuring the integrity of both the message and the 'signature' during transfer.

Message Verification on the Destination System

13. Once the 'signed' message and its accompanying 'signature' are on the target system, the verification process can be initiated:

```
bitcoin-cli   verifymessage   "_bitcoin_address"   "received_signature"
"chosen_message"
```

14. If the command returns 'true', it confirms that the 'received_signature' corresponds to the 'chosen_message' and was indeed 'signed' by the private key associated with the '_bitcoin_address'.

Matching the Address with an Online Source

15. To correlate the '_bitcoin_address' with an online source, one can utilise various online blockchain explorers. By inputting the Bitcoin address into the explorer's search bar, one can retrieve transaction histories, balances, and other associated data, verifying that the address is valid and active within the Bitcoin network.

[illegible]

Electrum Message Verification (Across Two Computers)

16. Electrum is a lightweight Bitcoin wallet that does not require users to download the entire blockchain. Notably, its functionalities encompass the 'signing' and verification of messages using Bitcoin addresses, akin to the Bitcoin Core software.

<u>Setting Up Electrum</u>

17. To set up Electrum the following steps should be undertaken:

    a. Downloading Electrum: Navigate to the official Electrum website at <u>https://electrum.org</u>. Ensure one is visiting the genuine site to avoid any counterfeit versions. Click on the 'Download' section and select the version that matches one's operating system.

    b. Installation: Once the download is complete, run the installer and follow the on-screen instructions. Launch Electrum upon completion and choose to create a new wallet or import an existing one.

18. This process should be undertaken for both computers (although it is not necessary to create or import a wallet on the second computer).

<u>'Signing' the Message</u>

19. To 'sign' a message, on the first computer, after setting up one's wallet and ensuring one's Bitcoin address is loaded, head to the 'Tools' menu and select 'Sign/verify message'. Input the message one intends to 'sign' and the relevant Bitcoin address from one's wallet. Click on 'Sign' to generate the 'digital signature'.

<u>Transferring the 'Signature' and Message to the Second System</u>

[illegible]

20. One can transmit the 'signed' message and its associated 'signature' from the first computer to the second computer via various means. This could involve sending an email, utilising a USB flash drive, utilising cloud storage, or even manually noting them down, ensuring both the message and the 'signature' remain unchanged in transit. In the relevant demonstration below, a USB flash drive was used.

<u>Verifying the Message on the Second Computer using Electrum</u>

21. To verify the message on the second computer, after launching Electrum, navigate to 'Tools' and select 'Sign/verify message'. Insert the Bitcoin address, the original message, and the received 'signature' into the respective fields. Press 'Verify'. If the software returns a confirmation, it attests that the message was 'signed' by the private key associated with the provided Bitcoin address.

<u>**Advantages of this Approach**</u>

22. The evident merit of the above methodologies is that the second system/computer, employed for verification, does not require possession of the private key. Thus, the risk of exposing one's private key is mitigated, especially if the verification process is conducted on a machine that is not secure. By decentralising the 'signing' and verification steps across two systems/computers, the integrity of the process is bolstered, and potential vulnerabilities are curtailed.

The demonstration to Rory Cellan-Jones and Ludwig Siegele

23. At paragraphs 208 to 212 of my first witness statement, I explained the meetings with the journalists, which took place in April 2016. While I cannot recall the demonstrations exactly, I have set out below the best of my recollection.

[illegible]

24. For each journalist, I recall demonstrating to them separately at least possession of the private key associated with block 9 using the Bitcoin Message Verification (Cross-System) method.

25. In particular, I recall using my Eurocom laptop especially configured to run on two separate operating systems, namely, a Windows 10 Enterprise system and a CentOS 7.2 (a Linux distribution) environment operating on a hypervisor virtual machine. This set up allowed shared access to the 'C:' partition by both operating systems, facilitating uniform file access and management across the operating systems. On this, I had downloaded the Bitcoin Core software on both operating systems. I verified this by using the cryptographic hash, which I address below.

26. First, I "signed" each said message with the text of a speech by Jean-Paul Sartre (the "Sartre.txt") using the private key for the Bitcoin address associated with block 9 of the Bitcoin blockchain on the Windows operating system (the source system).

27. In particular, on the Windows operating system (the source system), using the Bitcoin Core command-line interface, I 'signed' the message "Sartre.txt" using the private key for block 9, as follows:

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

28. This generated a digital 'signature', which cryptographically asserted that the message "Sartre.txt" was 'signed' using the private key corresponding to the Bitcoin address from block 9.

29. Once I had the 'signature' and the message, I was able to copy it across from the Windows operating system (the source system) to the CentOS operating system (the receiving system) on the same laptop. This was possible because I used a virtual machine for the Linux installation.

[illegible]

30. Upon receiving the Sartre.txt message and its 'digital signature' on the CentOS operating system (the receiving system), I used the following command on the Bitcoin Core software:

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

31. A return of "true" from this command affirmed that the provided 'signature' for the message "Sartre.txt" was genuine, and that the message had been 'signed' using the private key of the block 9 Bitcoin address.

The demonstration to Jon Matonis

32. At paragraphs 191 to 193 of my first witness statement, I explained my meeting with Jon Matonis. Here, I used the method described in paragraphs 23 to 31 above (the Bitcoin Message Verification (Cross-System) method). However, a different message was used and, in addition to the private key for the Bitcoin address associated with block 9, I also believe I used the method in relation to the private key for the Bitcoin address associated with block 11.

The demonstration to Gavin Andresen

33. At paragraphs 194 to 207 of my first witness statement, I explained my meeting with Gavin Andresen. My recollection is as follows: Following our conversation over email, Gavin wanted to travel to London. We met in a private room at a central London hotel, with Rob MacGregor and Stefan Matthews present. Following a lengthy discussion sharing insights into the creation of Bitcoin, its past, present and future as well as covering deeply personal territories, I agreed to use the 'digital signature' process to confirm possession of the early keys.

[illegible]

34. I was to use my own Lenovo laptop. For Gavin, Rob organised the purchase of a brand-new laptop from a retail store.

Computer Setup and Software Installation

35. I already had Electrum installed on my own laptop. This had been downloaded from Electrum's website: https://electrum.org. This laptop ran on the CentOS (Linux) operating system in a virtual machine.

36. Once Gavin's new laptop arrived, Gavin took the lead in setting it up from scratch. This necessitated an operating system installation. It ran a version of Windows, though I am not personally sure of the specific variant. My best estimate is that it was Windows 10.

37. Gavin was responsible for the setup. He first installed the Windows operating system, after which he connected the computer to the hotel's Wi-Fi network. Once the necessary updates and initial configurations for Windows were completed, Gavin proceeded to consider the appropriate Bitcoin wallet software for the machine.

38. Given that the Bitcoin Core software can often be time-consuming to synchronise with the entire blockchain, we discussed which software to use and my recollection is that Gavin opted for Electrum, a lightweight Bitcoin wallet that doesn't require synchronisation with the entire blockchain. 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, ensuring that the software hadn't been tampered with or corrupted. Following this, Gavin proceeded with the installation of Electrum on the machine.

39. I cannot, with certainty, specify the exact version of Electrum that Gavin installed. All I am sure of is that he sourced it directly from the official Electrum website, ensuring its authenticity and security.

[illegible]

<u>Message 'Signing' and Verification with Electrum Wallet</u>

40. I gave Gavin the choice to select any of the first 11 blocks for me to use for the demonstration. He selected blocks 1 and 9. In each case, I proceeded as follows:

a. On my laptop (the originating computer), I opened my Electrum wallet and navigated to the particular Bitcoin address selected by Gavin, possessing the private key with which I wished to 'sign' the message.
b. Under the 'Tools' tab, I selected 'Sign/Verify Message'.
c. In the dialog box that emerged, I typed in the message read out by Gavin (which I cannot now recall) in the 'Message' field.
d. Upon ensuring the message's accuracy with Gavin, I clicked on 'Sign'. Electrum prompted for a password as the wallet was encrypted. I entered this.
e. Once the message was 'signed' (or rather processed by the 'digital signature' algorithm), Electrum displayed the 'digital signature' in the 'Signature' box.
f. I copied and pasted this 'digital signature' to Windows notepad.exe and saved the file on a USB key which Gavin brought and provided to me. I cannot remember the name of the file each time, but it was something like "SignedMessage.txt File".
g. I safely ejected the USB key from my laptop (the originating computer) and gave it to Gavin who inserted it into his new laptop (the second computer). Gavin copied the 'SignedMessage.txt' file from the USB drive to the local storage of his new laptop (the second computer).
h. Gavin opened Electrum on his laptop (the second computer).
i. Gavin let me navigate to the 'Tools' tab and select 'Sign/Verify Message' on his laptop (the second computer) as he watched closely.
j. I typed the original message chosen by Gavin into the 'Message' box.
k. I then opened the 'SignedMessage.txt' file.
l. I copied the 'digital signature' from the text file into the 'Signature' box and clicked 'Verify'.

[illegible]

41. The first time in relation to the first block, this failed as I typed in the original message incorrectly. Gavin noticed that there was a typo and a word was misspelt. Once the typo was corrected, in each instance Electrum then confirmed the authenticity of the message.

Bitcoin Core Installation

42. On the Eurocom laptop used in the demonstrations above, I made sure that I adopted best practices when downloading and installing the Bitcoin Core software. I have described this below. I communicated this practice to the individuals involved in the demonstration so that they knew the process had been rigorously followed. The individuals could have validated the software installation themselves if they chose.

Secure Environment

43. I ensured that the Eurocom laptop's systems were protected with up-to-date security software, reducing the risks associated with potential malware or unwanted intrusions.

Software download

44. The Bitcoin Core software which I downloaded was, I believe, v0.11.2. This version was compatible with Windows and Linux-based operating systems. It was sourced from the official Bitcoin Core website. Avoiding third-party sources significantly mitigated the risk of encountering spurious or tampered versions. The software was subsequently downloaded and securely housed within the shared partition, denoted as 'C:'. Note that two downloads occurred as the Windows and Linux software uses separate install files.

Software verification

45. Before addressing the method of verification, it may be useful to explain why software verification important. The foundational purpose behind software verification is twofold: to ensure the authenticity and to validate the integrity of the software.

[illegible]

a. Authenticity: By verifying software, users are reassured that the software has been sourced from a legitimate and recognised origin, mitigating risks associated with counterfeit or maliciously disguised software.

b. Integrity: Beyond just verifying the source, software verification ensures that the software remains unaltered from its original state when produced by the software developers. This ensures that it's free from any post-production modifications which might introduce vulnerabilities or malicious functionalities.

<u>Cryptographic Hash Functions</u>

46. It may also be useful to briefly describe how software verification works. It is founded on the concept of cryptographic hash functions. A hash function is a mathematical algorithm that transforms any arbitrary data (often called the input) into a fixed-length series of bytes, typically manifesting as a string of alphanumeric characters. The resultant output, known as the hash value or hash digest, is uniquely representative of the given input. Two critical properties define these functions:

a. Deterministic: For a given input, the hash function will always produce the same hash value.

b. Avalanche Effect: Even the minutest alteration in the input will produce a dramatically different hash value, making any change, no matter how trivial, easily detectable.

<u>Hash Acquisition - SHA256</u>

47. CentOS has long been recognised for its commitment to system security and stability. Integral to its suite of security tools is the Secure Hash Algorithm (SHA), particularly the SHA-256 variant.

[illegible]

48. The SHA-256 algorithm belongs to the Secure Hash Algorithm 2 (SHA-2) family, a cryptographic hash function designed by the National Security Agency (NSA) and promulgated by the National Institute of Standards and Technology (NIST) in 2001. The numeral '256' in SHA-256 stands for the bit length of the hash output or digest. Specifically, the output is 256 bits or 32 bytes in length, culminating in a fixed-size, 64-character hexadecimal number.

49. Within the CentOS operating system, the 'coreutils' package furnishes a utility named 'sha256sum'. This utility empowers users to generate and verify SHA-256 cryptographic hashes. A quintessential application of this utility, especially when handling software like Bitcoin Core, is in the verification of the integrity of downloaded files.

50. To expand on this, when one issues the command 'sha256sum /path/to/file', the SHA-256 hash of the designated file is produced. This resultant hash can then be juxtaposed with a trusted hash value, typically proffered by software developers on their official platforms. Should the two hash values correspond, one can be imbued with confidence regarding the file's integrity, assured that no tampering has occurred since the creation of the trusted hash.

51. In software installations, especially those sourced from open-source repositories or direct online downloads, the verification of SHA-256 hashes is an important step which ensures that the software remains unaltered, neither by malicious intent nor inadvertent corruption during transmission. Given the gravitas of operations on platforms like Bitcoin where even the slightest aberration in the software can precipitate profound financial consequences the tools such as 'sha256sum' in CentOS assume a central role in ensuring software integrity.

[illegible]

Hash Verification on Windows

52. On the Windows 10 Enterprise platform on my Eurocom laptop, the in-built utility of Windows PowerShell was my tool of choice to verify the cryptographic hash. Windows PowerShell, with its robust set of cmdlets (command-let functions), provides an array of tools designed for system administration, including hash computation.

53. I ran the following command:

```
Get-FileHash C:\downloads\bitcoin_file_name -Algorithm SHA256
```

54. As a result, I was able to obtain the SHA-256 hash value of the Bitcoin Core software file situated within the 'C:' partition. This computed hash was then juxtaposed with the official SHA-256 hash, obtained from the Bitcoin Core website, confirming that they were the same. This confirmed the authenticity of the downloaded software and also allayed concerns about potential tampering or corruption during the download process.

Hash Verification on CentOS

55. For the CentOS 7.2 VM, I utilised the terminal and ran the following command on the file in the shared partition:

```
sha256sum /mount_path_to_C_downloads_on_CentOS/bitcoin_file_name
```

56. Again, the matching hash outputs from this and the official hash cemented the software's authenticity on the Linux environment.

[illegible]

Windows Installation

57. With the shared partition's utility, I directly ran the installer from 'C:' on the Windows 10 Enterprise system, navigating through the on-screen instructions completing a successful installation.

CentOS Installation

58. On the CentOS 7.2 VM, taking advantage of the shared access to the 'C:' partition, I extracted the downloaded tarball and commenced the installation process, ensuring Bitcoin Core was appropriately set up.

Connecting to the Bitcoin Network

59. Once the Bitcoin Core software was successfully installed on both systems, it was initiated to establish a connection to the Bitcoin network. The software began the process of synchronising with the network, downloading and verifying the entire blockchain history.

60. Upon completion of this synchronisation, the software indicated its up-to-date status, affirming that it had the most recent record of transactions and blocks from the Bitcoin network.

Operational Confirmation

61. With the software installed and synchronised:

    a. Status Check: I routinely checked Bitcoin Core's status to ensure it remained connected to the network and updated with the most recent blocks.

[illegible]

b. Backup and Maintenance: I set up periodic backups of the Bitcoin Core wallet and data, ensuring data safety. Furthermore, scheduled checks for software updates were put in place to ensure that Bitcoin Core was always running the latest, most secure version.

Conclusion

62. The installation of Bitcoin Core on the dual platforms completed without any difficulties. The use of the shared partition simplified the process. The detailed verification of software integrity ensured its authenticity and protected against potential tampering or corruption during the downloading phase.

Statement of Truth

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

Signed:	[illegible]
Name:	CRAIG WRIGHT
Dated:	09 OCT 2023.

[illegible]

Certificate of Compliance

I hereby certify that:

1. I am the relevant legal representative within the meaning of Practice Direction 57AC.
2. I am satisfied that the purpose and proper content of trial witness statements, and proper practice in relation to their preparation, including the witness confirmation required by paragraph 4.1 of Practice Direction 57AC, have been discussed with and explained to Dr Craig Wright.
3. I believe this trial witness statement complies with Practice Direction 57AC and paragraphs 18.1 and 18.2 of Practice Direction 32, and that it has been prepared in accordance with the Statement of Best Practice contained in the Appendix to Practice Direction 57AC.

Signed:	[illegible]
Name:	**Antony Craggs**
Position:	**Partner**
Dated:	**9 October 2023**