IN THE HIGH COURT OF JUSTICE **Claim no: IL-2021-000019**

BUSINESS AND PROPERTY COURTS OF ENGLAND & WALES

INTELLECTUAL PROPERTY LIST (ChD)

B E T W E E N:

CRYPTO OPEN PATENT ALLIANCE

Claimant

-and-

DR CRAIG STEVEN WRIGHT

Defendant

EXPERT REPORT OF ARTHUR ROSENDAHL

1

Chapter 1

Introduction

1.1 Background

1. I am Arthur Rosendahl, of Uppsala, Sweden, a software developer with over 25 years of experience in both using and programming LaTeX and the underlying TeX software (which is explained in section 1.3 below). I have a master's degree in mathematics (École normale supérieure, 2000 - 2005), and a bachelor's degree in physics (École normale supérieure, 2000 - 2001), and it was when I was a mathematics student that I first came across LaTeX. From that point I have been involved in some aspects of its development ever since, having co-developed several packages for LaTeX (polyglossia and hyph-utf8) and also being closely associated with the development of LuaTeX, a popular extension of LaTeX (which is also further explained in section 1.3 below). I am also a maintainer of XeTeX, one of the three major TeX engines. Generally, I am considered a TeX installation specialist, with further specialised knowledge of PDF files.

2. I am currently president of the TeX Users Group (TUG). TUG is a democratic, membership-based not-for-profit organisation founded to provide leadership for users of TeX and represents the interests of TeX users worldwide (and those are interested in typography and font design). TUG is responsible for supporting the development of TeX software, helping to run CTAN (the central repository of TeX packages and programs), and holding annual conferences in locations worldwide, amongst other things.

3. I was incidentally born on the year TUG was founded (1980), but my involvement with TeX started in 2005 at an annual conference in Wuhan. I have been on the board of TUG for over 10 years, the last six of those 10 being as vice president prior to my presidency in 2023. This, alongside my professional interest in LaTeX, has seen me participate regularly in TeX-based conferences. Although I was most active in these conferences in 2006-2013, I've been part of the organisation committee of the three online TUG conferences from

2020 to 2022 and am also active in other user groups of the TEX world, such as being the founding president of the ConTEXt Group, and a board member of GUTenberg, the French-language TEX users group.

4. Outside of LATEX I am general software developer, with my main programming languages being Ruby, Python, JavaScript (server and client-side) as well as C++. In terms of system administration and databases, I specialise in Linux, shell, both SQL and NoSQL database systems, and Docker. I currently work as a software developer and team lead for Uppsala University in Sweden (which I have held since April 2022), but have held numerous similar positions in both public and private institutions over the past 12 years. I make this Report in my own capacity. For further information, my CV is attached as {H/324} **Exhibit AR1** (noting that this is the most recent version I have in English, which is 3 years old).

1.2 Instructions

Duties and independence

5. 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 to my attention 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.

6. I can confirm that Dr Wright is not known to me personally or professionally, and I was not aware of the current proceedings prior to being contacted by Bird & Bird. I am not aware of any potential conflicts of interest in this case. I further confirm that the fees I am receiving are not dependent in any way on the outcome of these proceedings, and that the views expressed in this report have not been influenced by those fees in any way.

7. 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

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 White Paper, and whether or not Dr Wright is the pseudonymous creator, Satoshi Nakamoto, of Bitcoin and author of that paper. My instructions from Bird & Bird were split into two stages, which I answered sequentially. My answers for each stage are given in Chapters 2 - 3 of this report respectively. These staged instructions were as follows.

Stage 1

9. On 18 December 2023, Bird & Bird provided me with a PDF document which I was told was a copy of the original Bitcoin White Paper authored by Satoshi Nakamoto from 24 March 2009 (the "BWP", **Exhibit AR2**). I was asked to determine whether or not the BWP was generated in LaTeX, and to provide my observations.

{H/325}

10. I note that this analysis was done before I was aware of the nature of the arguments in the case, and therefore without knowledge of Dr Wright's documents. I further note that I was also provided with two further copies of the Bitcoin White Paper, one from 3 October 2008 (**Exhibit AR4**) and another from 11 November 2008 (**Exhibit AR3**). I understand these to be further copies of the White Paper which are also deemed, for the purposes of these proceedings, "control copies" of the Bitcoin White Paper.

{H/327}
{H/326}

Stage 2

11. On 22 December 2023, Bird & Bird then sent me a folder named "`TC`" which contained a number of different files (and file types), which included LaTeX source files, alongside a PDF file named "`Compiled WP.pdf`" which they understood were derived from one or more of the LaTeX source files that were contained within the `TC` folder and which was an attempt to produce an exact replica of the original BWP. Bird & Bird asked me to take a look at the contents of the folder and provide my observations as to: (i) what my impressions were about the sources themselves, how they were written, and the provenance of the code (as appropriate); (ii) whether I thought these sources contained the origin of the original Bitcoin White Paper from 24 March 2009 (via compiling a number of the sources or otherwise); and whether there were any differences or indications that I would need further information about to come to a conclusion.

12. Bird & Bird informed me that all the files within the `TC` were - and still are - subject to strict confidentiality terms. I was not to disclose the files in whole or in part to anyone, and I was to keep their contents confidential at all times. I was asked to, and have, stored these files securely and am in a position to destroy them upon request.

{E/23}

13. Bird & Bird then further provided me with a document titled "CSW8.pdf", which was a witness statement from Dr Wright, the person who I understand created the files within

the TC folder. Bird & Bird explained to me that this statement was supposed to explain the computing environment which includes the software, compiling engine, all packages specified in the code, and all relevant versions of the foregoing, sufficient to allow one to reproduce the BWP under the conditions specified. They stated that this may prove useful in aiding my analysis, and asked me to provide any observations I had on this computing environment and whether it provided the necessary technical conditions to reproduce the BWP from the LaTeX source files.

14. Any other documents that were provided to me by Bird & Bird throughout my instruction have been mentioned throughout the course of my report.

1.3 Introduction to LaTeX and terminology

15. Because LaTeX[1] and related computer programs are a central part of this analysis, I need to spend some time explaining what they are. LaTeX is a typesetting system that was created in 1984 and has been in continuous development since. It builds on the program "TeX", which first released in 1978. The latter forms a bottom layer that is usually referred to as the *engine*, and has also seen many new developments since first coming into existence. TeX is a programming language, meaning it can be enhanced in all sorts of ways, and indeed has been over the years.

16. LaTeX is the top layer of that duality and comes with a set of core commands that allow formatting, mathematical typesetting, a system for managing bibliographic and other references, and other document controls. Most users of LaTeX rarely interact with TeX itself.

17. By far the most common use of LaTeX is for creating scientific documents, with mathematical formulae, tables, etc. It is in this sense a sort of word processor for the scientific community, with the essential difference to something like Microsoft Word being that documents must be first input as source code, then compiled into the final result using a TeX engine (which nowadays is almost always a PDF file). During this process, additional files are created, most importantly a log file that records the compilation, as well as an auxiliary - or `.aux` file - that stores additional information about the structure of the document.

18. LaTeX also makes use of a number of document *templates*, which makes it very easy to create well-typeset documents automatically, without needing to be concerned about formatting. For example, when writing an academic article the "article" template can be specified, and the resulting format will be correct for the intended publication; while someone writing a letter could use the "letter" template. The content of those documents

[1] LaTeX is generally pronounced as "Lay-Tek" or "Lah-Tek", as if with a hard "ch" sound, and the associated programs similarly. XeLaTeX, which will be introduced shortly, is pronounced "Zee-Tek".

can then be written in plain text, with a few commands to specify information such as the title, author details, and other formatting.

19. The core functionality of LaTeX can also be extended further, by using many available add-ons, known as *packages*. These are software modules which provide additional functionality. These are invoked by calling on them using the command `\usepackage`, at which point whatever commands or options are included with the package become available for use in the document. That could be seen as a third layer, but I think the better description is that they give "breadth", adding functionality and expressivity to the underlying language. Packages can take almost any size, ranging from just a few lines of code to being extremely complex additions. Many of them have different options that affect their behaviour, and they do not always work very well with each other; errors arising out of packages being loaded "in the wrong order" is a common source of frustration for all LaTeX users. I will say more later on about the packages that are relevant for this instruction. For the past three decades, packages have been collected in the Comprehensive TeX Archive Network, or CTAN, created in 1992. It contains, I am told, over 5000 packages.

20. To give an example, this report is written using the default "report" template. I have used the core command `\linespread` to widen the gaps between lines. I have also made use of a few packages, with commands that allow me to number every paragraph sequentially, and used the package `geometry` which allows me to specify the width of each margin.

21. TeX *distributions* are software applications which are typically installed on a user's computer. They will bundle together the various components needed for a working TeX system, allowing documents to be written and compiled. Up until quite recently, LaTeX was acquired on Windows by installing one of the two major *TeX distributions*: TeX Live or MiKTeX, which provide editing software that can be locally installed and used by anyone. This changed about a decade ago with the advent of Web services running LaTeX "in the cloud", a landscape now dominated by the London-based company Overleaf. The Overleaf service runs on TeX Live, an extensive distribution that in 2023 contained more than 230,000 files when installed on a computer. TeX Live takes in turn its packages from CTAN.

22. As might be expected from any software that has stood the test of time, LaTeX and the accompanying systems have changed a lot since their inception. One possibly surprising fact is that some of the most significant developments have occurred in the *engine*, the bottom layer. Over the years, many extensions have been written to the original TeX program, the most relevant ones for this analysis being, in chronological order of their release:

    a. pdfTeX, released in 1997, was the first TeX engine to produce PDF files directly,

without recourse to external converter tools;

b. XeTeX, released in 2004 on Mac OS and 2006 on Linux and Windows, was the first TeX engine to support the character encoding standard "Unicode", and to use most font formats (released in 2004 on Mac OS and 2006 on Linux and Windows); and

c. LuaTeX, released in 2006, contained the embedded language Lua, with somewhat similar aims to XeTeX. Lua itself is its own programming language, entirely separate to TeX. With LuaTeX, it is possible to use both languages together in conjunction, and compile them in one document. This would not be possible with other engines, which would not understand how to interpret Lua code.

23. LaTeX, the top layer, has of course changed considerably over time too, but these variations are not as easily identifiable. When used on top of pdfTeX, LaTeX is often referred to as pdfLaTeX, and likewise we have XeLaTeX and LuaLaTeX. The difference between e.g. LuaTeX and LuaLaTeX does usually not matter to the end user, as both layers have to be used simultaneously.

24. Since I just alluded to font formats, I should mention that before XeTeX came into being in 2004, using custom fonts with any TeX systems was a rather complicated affair and involved creating a number of ancillary files; changing the maths fonts, in particular, was fiendishly difficult – and still is, to some extent. Fonts also come in a number of different formats, such as TrueType and OpenType, among others: the most widely used font format with TeX in the 1980s, 1990s, and even into the 2000s, was called "Type 1", originally developed by the software company Adobe. pdfTeX changed the situation, partly by making it somewhat easier to use the more common TrueType format, and XeTeX and LuaTeX then brought yet further changes and eventually full support of TrueType and the newer OpenType font format. In parallel, LaTeX has always had a rich font machinery, where individual font faces can be linked together in families, and the user can switch within a family with simple commands.

25. As with any programming language, TeX can include comments: pieces of code that are ignored by the compiler and that will not be typeset into the compiled document. The comment character in TeX is the percent sign: anything after '%' on a line will be considered a "comment", which could be an actual comment on the code, but could also be used as a way to suppress, or *comment out*, a chunk of the code that was not needed. Both uses of the comment function are very common, on par with normal practice in any programming language, where parts of the code are regularly taken out and put back in, in a trial-and-error process.

26. LaTeX is open-source, in that all the source code is freely available, and free of charge for anyone to view, use, or modify. All the major TeX engines, and most extant LaTeX

packages, are likewise open source. This has the benefit of exposing the software source code, including changes over time, in maintained repositories. This allows inspection of their functionality and how it changed over the years, and I will make extensive use of that fact when it comes to discussing the history of the different packages that are relevant in this case.

27. Finally, I apologise unreservedly for using the traditional typesetting of the name TEX and its derivatives, that – in the words of an old friend of mine from the community – makes any publication about TEX look like a high school magazine. I do not know how else to refer to these names, as I do not consider that typing "TeX" or "LaTeX" is necessarily better. Attempts to normalise the spelling have never caught on.

Chapter 2

Analysis of the Bitcoin White Paper

28. To reiterate my Stage 1 instructions, on 18 December 2023, Bird & Bird instructed me that they needed to ascertain whether or not a specific PDF document was generated in LaTeX. They informed me that I would also need to look at some LaTeX source files on confidential terms, but that they had not yet been provided with them, and so asked me to begin work by analysing the PDF first. The PDF document they then sent me for analysis was a copy of the Bitcoin White Paper by Satoshi Nakamoto dated 24 March 2009. I made a number of observations. I note again that this analysis was done before I was aware of the nature of the arguments in the case, and therefore without knowledge of Dr Wright's documents, though I was later provided with them as I explain further below in section 3.

2.1 Typography

Overall presentation

29. I started by taking an overall look at the document's presentation. The general look and feel of the BWP does remind one of LaTeX, in particular the placement of the title, the information about the author, the abstract, and most importantly the numbered sections and the list of references and their formatting at the end all give a general impression that the document could have been created in LaTeX. The presence of a few formulae, diagrams, and an enumeration add to that impression.

30. There are however a number of odd details that deviate from the usual appearance of a LaTeX document, including:

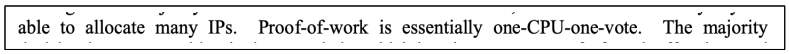


Figure 2.1: Example of overstretched word space in the BWP

a. Throughout, the apostrophe is straight (') instead of curly (’). Fonts are generally set up in LaTeX to use the curly apostrophe, even when a straight quote is input in the source code;

b. In the BWP section numbers are followed by a full stop, whereas in the LaTeX default they are not;

c. The formula at the bottom of page 4 of the BWP uses the character ‘*’ for multiplication instead of $\times$. A LaTeX user would likely have used “maths mode” to achieve something looking like: $80\ \textit{bytes} \times 6 \times 24 \times 365 = 4.21\,\text{MB}$;

d. In the enumeration at the beginning of section 5 of the BWP, the numbers are followed by a closing parenthesis as in: *“1) New transactions are broadcast to all nodes”*, whereas the LaTeX default is no parenthesis;

e. Although the text is justified-aligned (flush straight at both the left and right margins), there is no hyphenation (word division across line breaks). As a result of not using hyphenation, the inter-word space in the BWP is stretched a lot in some places, as in figure 2.1, that appears in the middle of page 3 of the BWP.

   LaTeX is set up by default to allow some (but not too many) words to break across lines, resulting in more even spacing. Hyphenation is an important part of how LaTeX achieves good typesetting in documents that are justified-aligned, without stretching the inter-word spacing. It is possible to deactivate hyphenation but the result is generally considered inferior, as most users who follow that route find out;

f. The formulae are not centred, whereas in LaTeX they are; and

g. The formulae also look a bit awkward in places, as for example with the uneven spacing around the fraction bar in $(q/p)^z$ at the bottom of page 6 of the BWP, as well as in two places in the middle of page 7. See figure 2.2 for a comparison the white paper with LaTeX.

Choice of Fonts

31. I also need to remark upon the choice of fonts. The main text is in Times New Roman, which is indeed used very often in scientific articles; alongside with the LaTeX default, Computer Modern Roman; but the white paper’s title and section headings are in a different font, Century Schoolbook. It is uncommon in LaTeX to use different fonts for the text body and the headings. Also, the code extracts on pages 7 and 8 are in Courier,

$$q_z = \begin{cases} 1 & if\ p \leq q \\ (q/p)^z & if\ p > q \end{cases}$$

$$q_z = \left\{ \begin{array}{ll} 1 & if\ p \leq q \\ (q/p)^z & if\ p > q \end{array} \right\}$$

Figure 2.2: First formula of the Bitcoin white paper, and its rendering in LaTeX

a very thin typeface that doesn't render very well. I would have expected the default LaTeX monospaced font here, Computer Modern Typewriter. The Courier font, while a default font for Windows, is not a default font for LaTeX and would have required a greater degree of effort to use, while Computer Modern Typewriter could be used simply with the core LaTeX command `\texttt`.

32. There is of course no accounting for taste (and this was only the starting point of my review), but the overall impression I gained was that, if the document had been produced with LaTeX, there has been great attention to details in some areas, and apparent carelessness in others: many of LaTeX's default settings would have to have been changed, not necessarily for the better, in ways that would have taken effort to achieve. I already commented on spacing above, and can also point to the mathematical formulae. Regardless of which fonts one likes best, I think most people would agree that the second example in figure 2.2, made in LaTeX with standard settings, look at least a little better than the first one and is certainly at least as good. Changing the default settings to achieve the exact spacing observed in the formula on top seems practically infeasible to me; if it could be done, it would most likely have to be done at the lower level, in TeX instead of LaTeX, and take substantial extra time. It may be possible, using specific commands, to input additional spaces manually to achieve a similar spacing, but that would not be guaranteed to have the same result, and it would still be necessary to typeset the symbols differently.

Format of embedded fonts

33. I also made some observations about the format (i.e. the filetype) of the font files embedded within the PDF file. When a PDF is output, it is common for copies of its required fonts to be embedded within the structure of the document: this enables the PDF to be displayed on a range of systems, without making assumptions about which fonts are installed locally. The program "`pdffonts`"[1] gives a summary of what fonts are present in a PDF file. For the original Bitcoin White Paper, its output is shown in table

[1] See https://www.xpdfreader.com/pdffonts-man.html

name	type	encoding	emb	sub	uni	object ID
BAAAAA+CenturySchoolbook-Bold	TrueType	WinAnsi	yes	yes	yes	33 0
CAAAAA+TimesNewRomanPSMT	TrueType	WinAnsi	yes	yes	yes	53 0
DAAAAA+TimesNewRomanPS-BoldMT	TrueType	WinAnsi	yes	yes	yes	63 0
EAAAAA+ArialMT	TrueType	WinAnsi	yes	yes	yes	38 0
FAAAAA+TimesNewRomanPS-ItalicMT	TrueType	WinAnsi	yes	yes	yes	58 0
GAAAAA+OpenSymbol	TrueType	WinAnsi	yes	yes	yes	43 0
HAAAAA+CourierNewPSMT	TrueType	WinAnsi	yes	yes	yes	48 0

Table 2.1: Fonts from the Bitcoin White Paper

2.1.

34. As previously observed, the main fonts of the documents are indeed Times New Roman, Century Schoolbook, and Courier, but these exact names are actually not what one would expect in a LaTeX document, because many commonly used fonts are actually open source "clones" of existing proprietary fonts and therefore do not use the proprietary font files or font names themselves. In a typical LaTeX document, the font mimicking the visual appearance of Times New Roman would be called either "Nimbus Roman N° 9L", or "TeX Gyre Termes", which would have then showed up in table 2.1. Similarly, the equivalent font that can be used to replace Century Schoolbook is "TeX Gyre Schola". The presence of these commercial font names in PDF file produced by LaTeX, rather than their equivalent from the TeX world is very unexpected; I can't remember observing it before.

35. The font of the mathematical formulae within the BWP is, like the main text font, Times New Roman. In March 2009 this would have been a rare choice had the BWP been created in LaTeX, if not entirely impossible. There were back then very few other options for maths mode than the default font, Computer Modern; Times was indeed one of them, but in my early attempts at reproducing reproducing the White Paper's formulae in LaTeX, I was not able to achieve exactly the same result. The Greek lambda was not the same, and neither was the Latin 'z', oddly. I experimented with trying to reproduce these characters in different ways and found that using slightly different settings, I was able to at least change the shape of the lambda to a more slanted one, but even then that still didn't match the one in the BWP.

2.2 A deeper dive into the PDF structure of the Bitcoin White Paper

2.2.1 Introduction to the general structure of PDF format files

36. At this point, an introduction to the inner workings of a PDF file is in order. This well-known acronym stands for "Portable Document Format", created by Adobe in 1993, and took inspiration from the earlier PostScript, also by Adobe, which was a fully-fledged

programming language focused on describing and producing printable documents.

37. At the lowest level, a PDF file encodes a collection of *objects* describing the fonts, images, the contents of each page, other graphical elements and information about the placement of each of these things on the page), as well as some metadata. These objects are tied together by a number of structural elements, one of which, appearing at the very end of the file, is called the *trailer* and contains technical metadata that is not usually exposed to the user viewing a PDF document. there is also a similar *header* section at the beginning. Objects are identified by an *object number*, a positive integer incremented starting from 1, as well as a *generation number*, usually 0. The latter was originally envisioned as a version number, with PDF files containing several different versions of the same object and the trailer directing which one should be used. To my knowledge this possibility has rarely been used, if ever (if an object needs to be updated, the whole file is regenerated), and I only mention it because generation numbers can still be seen in object IDs today, as in figure 2.1.

38. Objects usually start with a list of a key-value pair, to which is often appended a *stream*, a sequence of bytes whose meaning depends on which object they belong to. The contents of each page is coded in the stream of an object, usually in a compressed form which can be readily decompressed using a number of tools. Figure 2.3 shows a few objects extracted from the original BWP. I show them in the order in which they appear in the file, but for a clearer understanding we need to start at the bottom: there we see object number 66, the PDF file's *"root object"*, that forms the top of the hierarchy of objects.

39. The root object refers, amongst other things, to another object that gives the full list of pages: that's object number 28. We know that because of the entry `/Pages 28 0 R` where `/Pages` is the key and `28 0 R` is the value. In this case, the value itself is just a reference to object number 28. Object 28 contains, in its `/MediaBox` key, the dimensions of the pages (595pt × 842pt, which is standard A4 size)[2], a reference to a list of "resources", the number of pages (9), and most importantly, a list of pages: the value of the key `/Kids`. The first of these pages is object number 1, that has a reference to another object describing its contents, number 2.

40. That object number 2 happens to be the very first object in the PDF file. The most interesting part of it is the binary stream: it contains, once uncompressed, the full description of page 1 of the BWP. The key-value pair `/Filter/FlateDecode` tells us that it is compressed using the "flate" filter (a pun on "deflate"), the most common compression method for page content streams. An interesting tidbit is that the length of the content stream is indicated via a reference (to object number 3): it is common to do so, so that the program writing the PDF file byte-by-byte does not need to know the

[2]This is inconsistent with the dimensions given for the individual pages, that all have a media box of `[0 0 612 792]` or U.S. letter. It can be seen in object 1 in my example. I came across this oddity a few days before handing over this report and do not know what to make of it. It may be a bug in the software that produced the PDF file.

length of the stream in advance. It can output it to the file, keeping track of how many bytes it writes, and then store the length separately in a different object. The length of an object's stream is a mandatory part of any object that has a stream. Not all do, as can be seen in the examples I show. I will give later relevant extracts of the page content stream of PDF files, decompressed.

Image encoding within PDF files

41. There are two main ways that images can be embedded within PDF files. One way is to use can be either bitmap formats such as BMP, JPEG, PNG, GIF or other common formats, which encode rectangles of pixels. The other way is to encode a series of lines and curves, called *vector* graphics. A major benefit of vector graphics is that they can be scaled to any dimension without loss of resolution.

Font encoding in PDF files

42. PDF supports a number of font formats, most notably Type 1 – invented by Adobe themselves – and TrueType – invented by their main competitors as typographic software vendors in the 1980s and 1990s, Apple and Microsoft. When included in a PDF file, however, the terms "Type 1" and "TrueType" mean something slightly different than when applied to standalone font files that might reside on a computer; the only relevant difference for us is that these font types, due to the way they are embedded and stored in the PDF format, can only have up to 256 glyphs (where a glyph means a single representation of a character), selected from a table of glyphs by using single-byte character codes (which can take a value from 0 to 255). These types are called *simple fonts* in the PDF specification. The other types of fonts, *composite fonts*, are encoded with two-byte codes (allowing support for a greater number of glyphs) and are identified by a type starting with "CIDFontType" as PDF objects.

43. The vagaries of the two competing font types have been dubbed by some the "font wars" and led to the ultimately successful efforts to unify the different font format under the umbrella of *OpenType*, whose technical specification was first published in 1997. However, still today and certainly at the time the Bitcoin White Paper was created in 2008-2009, OpenType would never be identified as such in a PDF file, but would be extracted as either Type 1, TrueType, or one of the CID font subtypes.

2.2.2 The innards of the Bitcoin White Paper itself

44. Equipped with this knowledge, we can now take a further look inside the PDF file of the BWP itself.

```
%PDF-1.4
%Ã¤Ã¼Ã¶ÃŸ+
2 0 obj
<</Length 3 0 R/Filter/FlateDecode>>
stream
(4448 binary bytes omitted)
endstream
endobj

3 0 obj
4448
endobj
...
1 0 obj
<</Type/Page/Parent 28 0 R/Resources 65 0 R/MediaBox[0 0 612 792]
/Group<</S/Transparency
/CS/DeviceRGB/I true>>/Contents 2 0 R>>
endobj
...
28 0 obj
<</Type/Pages
/Resources 65 0 R
/MediaBox[ 0 0 595 842 ]
/Kids[ 1 0 R 4 0 R 7 0 R 10 0 R 13 0 R 16 0 R 19 0 R 22 0 R 25 0 R ]
/Count 9>>
endobj

66 0 obj
<</Type/Catalog/Pages 28 0 R
/OpenAction[1 0 R /XYZ null null 0]
/Lang(en-GB)
endobj
...
```

Figure 2.3: Extracts from the PDF file of the Bitcoin White Paper

Metadata entries in the BWP

45. Examining the metadata of the Bitcoin White Paper, it clearly states that it has been created by OpenOffice.org 2.4. This can be seen in most PDF viewers, with a function called "show info" or something similar, as shown also in figure 2.4. The BWP identifies its "producer" as "OpenOffice.org 2.4" and its "creator" as "Writer" – Writer is the word processor from the OpenOffice software suite. I noted that 2.4 was indeed the current version in 2009, with all releases happening either that year or the year before (2.4.0 to 2.4.3)[3].

46. However, that metadata (though a helpful indication) is not entirely reliable, since it is possible to change the metadata when working with LaTeX (see 3.7.2 in the next chapter). Nonetheless, in my view the fact that the BWP states it has been made with OpenOffice is significant.

Naming of fonts embedded in the BWP

47. In the output from `pdffonts` shown in table 2.1 earlier, the recognisable names of the fonts are prefixed by seemingly arbitrary strings of uppercase letters. The reason for this is that the fonts are not written wholly into the PDF file, but are included as subsets in order to gain space and to make it harder to extract a full font file from the PDF file, which is otherwise rather straightforward. It can also happen that the same font is included twice in the PDF, as two different subsets. In order to preserve the distinction between the two subsets, a 6-letter string, followed by the character '`+`', is prefixed to subset font names to designate them. When TeX engines generate these names, the 6-letter designations would have been chosen randomly; however, in the white paper, they are chosen in a predictable manner: `BAAAAA`, `CAAAAA`, etc. This means that the BWP, had it been created with LaTeX, could only have been produced by subtly modifying a version of a TeX engine at a low level, so as to output these deterministic prefixes of how fonts are labelled.

48. That output in the BWP is, however, consistent with how fonts are labelled when converting to PDF within OpenOffice, which is consistent with the metadata recorded above.

Types of embedded fonts in the BWP

49. Returning to the issue of the fonts, I can now discuss their type, displayed in the second column of table 2.1: it is `TrueType` for all fonts. This is a strong indicator that, if some TeX-based system had been used to create the Bitcoin White Paper, the engines XeTeX and LuaTeX are very unlikely to be have been involved, because they do not use the PDF font type `TrueType` even when including `TrueType` fonts. I realise this statement may sound absurd and so will unpack it: because those engines are aimed at supporting

[3] See `https://wiki.openoffice.org/wiki/Product_Release`.

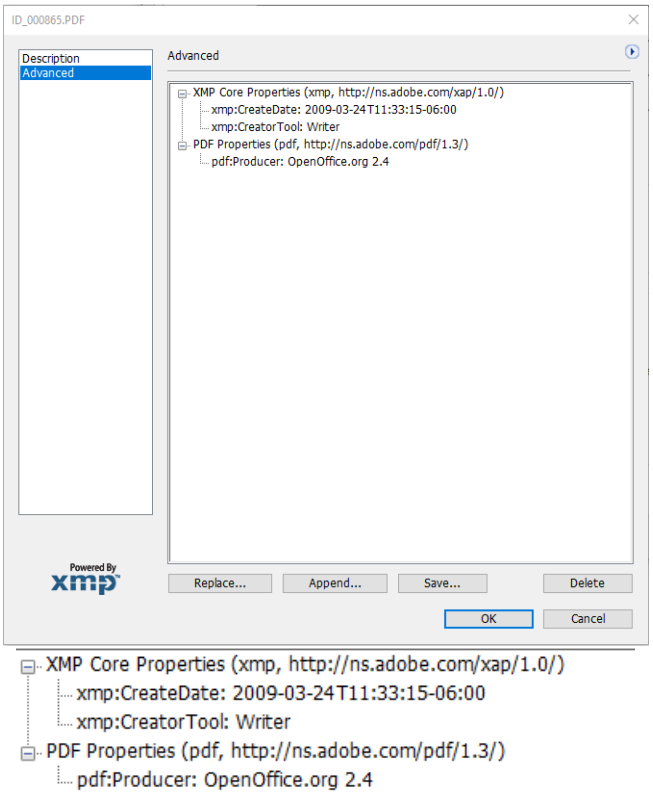


Figure 2.4: The metadata of the Bitcoin White Paper, as seen in Adobe Acrobat (as well as a magnified view)

```
134.3 684.4 Td
/F1 14 Tf
[<01>-2<02>-1<03>4<04>-1<05>-4<02>-1<06>-1<07>-2<08>1<09>1<08>1<0A>1<0B>-5
<0B>3<0C>-2 <0D>-3<03>4<05>-4<0D>4<0A>1<0B>-5<0B>3<0C>-2<08>1<0E>1<0F>-5<0B>
3<04>6<03>-4<0C>-2 <05>-4<06>6<02>-1<04>-1<08>1<10>-2<11>-4<1213>-1<08>1<14>
2<15>-4<12>7<03>-4<0B>3<16>]TJ
```

Figure 2.5: Page content stream from the Bitcoin white paper

a wide range of characters, they use the "composite fonts" I introduced in section 2.2, resulting in a font type of `CIDFontType0` or `CIDFontType0C` for most font files. The exception was Type 1 fonts, which were already supported well and would be included as `Type1` font objects in the PDF file. It was already possible to use TrueType fonts with LaTeX before XeTeX and LuaTeX came along, if the engine pdfTeX was used. In that case the font files would have indeed been embedded in the PDF file with a type of `TrueType`. Considering alternative engines, some TeX engines before pdfTeX could be tortured into accepting TrueType fonts, but they are even less likely candidates for an engine that could be used to create this document. Therefore, if a TeX engine had been used at all to generate the Bitcoin White Paper, it would have been via pdfTeX, not XeTeX, LuaTeX, or any other TeX engine.

50. While the embedded fonts do not correspond to the output expected of any TeX engine, I note that OpenOffice does embed fonts in this way, and the font embeddings are therefore also consistent with the recorded metadata showing OpenOffice as the editing software used.

Page content stream and text encoding in the BWP

51. Let us now move on to the page content streams within the BWP. In figure 2.5 I show an extract from the first page of the contents stream of the White Paper, with line breaks that I added myself.

52. The first line uses the `Td` operator, which is an instruction to move to the point whose coordinates are specified just before it. On the second line, the `Tf` operator is an instruction to set the current font: that's referred to as `F1`, defined as an object in the file (it happens to be Century Schoolbook Bold). The number 14 is the point size. Finally comes the main part of the content stream: using the `TJ` operator to set text at the current point of the page, it specifies an array of strings enclosed by `<>`, interspersed with numbers. The strings are encoded in hexadecimal (base 16), meaning that two hexadecimal characters encode exactly one byte ($16^2 = 2^8$). The numbers specify the distance in which to move the current point after having displayed the string on the page, and are typically rather small (the measurement units for these numbers are set by the font). In other words, what happens here is that the PDF interpreter will first

```
/F39 9.9626 Tf
0 -17.932 Td
[(W)80(e)-269(consider)-270(the)-269(scenario)-269(of)-269(an)-270(attac)
1(k)10(er)-270(trying)-269(to)-269(generate)-269(an)-270(alternate)-269
(chain)-269(f)10(aster)]TJ
```

Figure 2.6: Page content stream from pdfTEX output

see the string `<01>`, set it at the current point of the page, then move forward by 2 small steps, then set the string `<02>`, etc. The strings encode each time a single character of the font, with one exception near the end of the excerpt (`<1213>`).

53. A similar extract from a document created with pdfTEX, shown in figure 2.6, exhibits some significant differences to the method used to encode the text including:

    a. The font is selected before the current point is set;

    b. The strings are encoded in a different way (using letters instead of hexadecimal characters); and

    c. They are also much longer.

54. Of those points, the last point is the most significant: whereas in the original Bitcoin White Paper, characters are mostly written into the PDF file one by one, here with pdfTEX longer chunks are written at once. I can explain that by the way TEX engines process the text to be typeset: once the input text has been read, TEX builds a chain of *nodes* that can be either printable characters, or *glue*. Glue is the not very apt name for space that can stretch or shrink (the creator of TEX himself admitted that "spring" would have been a better name). All the spaces between words are converted into glue nodes during the TEX run. In addition, some glue may also appear in the middle of words because of the process of *kerning*, which means making small space adjustments between pairs of adjacent characters to balance their visual appearance on the page. A common example is that the characters 'A' and 'V' have to be kerned in most fonts, as they would otherwise look too far apart due to their complementary slopes: compare AV and AV (the latter unkerned). This kerning, and fine control of spacing with the concept of glue, is an important characteristic of TEX-based systems.

55. In figure 2.6, kerning as used by pdfTEX is shown by the numbers outside the parentheses: 80 between 'W' and 'e' on the first line, 1 between 'attac' and 'k', then 10 between 'k' and 'er' on the second line, etc. As can be seen, there are relatively few instances of kerning inside words. By contrast in figure 2.5 it can be seen that the BWP is encoded with almost all character pairs separated by a small explicit space. For TEX to produce such output would mean that every single pair of characters would be need to be kerned in a font: this is extremely unlikely, meaning that the output we see in figure 2.5 is in

```
0 0 0 rg
229.5 193.5 m
222.7 191.2 l 222.7 195.8 l 229.5 193.5 l h
f*
0 0 0 RG
189.8 193.5 m
224 193.5 l S
0 0 0 rg
337.2 187.8 m
330.4 185.5 l 330.4 190.1 l 337.2 187.8 l h
f*
0 0 0 RG
```

Figure 2.7: Path construction operators in PDF, extract from page 1 of BWP

turn extremely unlikely to have been produced by a standard TEX setup.[4]

56. This is, however, how OpenOffice typesets information in a PDF, and again is therefore consistent with the displayed metadata.

Diagrams in the BWP

57. I was asked by Bird & Bird whether I had any comments on the diagrams. I had inspected these but it did not inform my views. I noted that they were coded using PDF path construction operators, that is to say as a set of lines, geometrical figures, and text inside the page content stream of each page where they appear. It looks like the example on figure 2.7 and I cannot draw any definite conclusions either way: it could have been produced either in LATEX with the help of some package, or by a different program.

Header and Trailer in the BWP also indicate use of OpenOffice

58. I also noticed two points of information in the header and trailer of the PDF file.

59. Starting with the trailer, as shown in figure 2.9: I note that it has a piece of information I would not normally expect to see in a PDF: the element `/DocChecksum`. That looks a little different from the other pieces of data. I looked for an explanation of what in was and could not find a mention of it in the technical specifications of the PDF format [5].

60. I came to realise it was not part of the standard specification, but was something unspecified and unique to OpenOffice; it is not a standardised field and is not output

[4] I also note that if this had been produced by a TEX engine, it could only have been an engine which uses single-byte (8-bit) fonts, which would exclude the use of XETEX or LuaTEX, since those output 16-bit fonts.

[5] I looked at three different versions of the standard: 1.4, 1.6, and 2.0, which covers releases from 2001 onwards.

```
+    // prepare document checksum
+    OStringBuffer aDocChecksum( 2*RTL_DIGEST_LENGTH_MD5+1 );
+    if( m_aDocDigest )
+    {
+        sal_uInt8 nMD5Sum[ RTL_DIGEST_LENGTH_MD5 ];
+        rtl_digest_getMD5( m_aDocDigest, nMD5Sum, sizeof(nMD5Sum) );
+        for( unsigned int i = 0; i < RTL_DIGEST_LENGTH_MD5; i++ )
+            appendHex( nMD5Sum[i], aDocChecksum );
+    }
     // document id set in setDocInfo method
     // emit trailer
     aLine.setLength( 0 );
@@ -5041,6 +5079,25 @@ bool PDFWriterImpl::emitTrailer()
         aLine.append( m_aDocID.getStr(), m_aDocID.getLength() );
         aLine.append( "> ]\n" );
     }
+    if( aDocChecksum.getLength() )
+    {
+        aLine.append( "/DocChecksum /" );
+        aLine.append( aDocChecksum );
+        aLine.append( "\n" );
+    }
```

Figure 2.8: Excerpt of commit dated Mon Mar 26 10:21:15 2007 +0000, number d217c079d7b3ca7b5039428594e7cdfdf9a0c4a9, showing introduction of DocCheckSum to OpenOffice. Note the use of green text with + designators indicating new lines added.

by any other PDF producer. It is possible to identify this within the source code of OpenOffice itself, and at the web page `https://git.libreoffice.org/core/+/d217c079d7b3ca7b5039428594e7cdfdf9a0c4a9%5E%21` to inspect for the "commit" that introduces this keyword into the source. The relevant section of the commit is also extracted in figure 2.8[6]. Although this is not strictly speaking conformant to the PDF specification, it seems that most PDF viewers are able to ignore that additional piece of information which is meaningless to them. Regardless, its presence in the BWP is very significant: it means that the BWP can only have been produced by OpenOffice or one of its successor programs[7], or a program modified specifically to mimic its output. The latter could not be done in LaTeX directly and would entail modifying the underlying engine at a low level. It is theoretically possible, but it is not a small task: the person attempting to do so would need to know TeX's code base extremely well and have some knowledge of the PDF internals too. I cannot imagine any reason for wanting to do so, and I do not think it plausible.

61. After I discussed the above findings with Bird & Bird, I was shown an article titled *Robust PDF Files Forensics Using Coding Style* by Adhatarao and Lauradoux, which is exhibited to this Report as **Exhibit AR5** and is available at `https://arxiv.org/pdf/` {H/328}

[6] And see also `https://bugs.documentfoundation.org/show_bug.cgi?id=66580` for a discussion of its presence.

[7] That would be LibreOffice, a fork of OpenOffice created in 2010, which is based on the same underlying source code as pre-2010 OpenOffice.

```
<</Size 68/Root 66 0 R
/Info 67 0 R
/ID [ <CA1B0A44BD542453BEF918FFCD46DC04>
<CA1B0A44BD542453BEF918FFCD46DC04> ]
/DocChecksum /6F72EA7514DFAD23FABCC7A550021AF7
>>
```

Figure 2.9: PDF trailer of the original Bitcoin White Paper

> ***Note:*** *If a PDF file contains binary data, as most do (see Section 3.1, "Lexical Conventions"), it is recommended that the header line be immediately followed by a comment line containing at least four binary characters—that is, characters whose codes are 128 or greater. This will ensure proper behavior of file transfer applications that inspect data near the beginning of a file to determine whether to treat the file's contents as text or as binary.*

Figure 2.10: Comment from PDF specification version 1.4 indicating the purpose of the binary bytes encoded within in line 2 of a PDF

`2103.02702.pdf`. This contains a discussion of certain technical indications allowing identification of PDF-producer programs, and in particular includes the trailer and the `/DocChecksum` point that I observed above. At page 6 of the article, there is a table showing different markers of PDF producers.

62. The article named another interesting way to identify PDF-producing programs, namely by inspecting the second line of the PDF file. The first line of a PDF file always consists of: the percent character (which is a comment marker), followed by the string `PDF-`, followed by the version of the PDF format the file conforms to. After that, the second line is a sequence of arbitrary binary bytes chosen by the particular program that output the PDF file. According to a note within the PDF specification, extracted here as figure 2.10, the purpose of these bytes is to help convince any applications inspecting the file that it contains binary data (and not, say, plain text). The PDF specification does not strictly require this, and does not lay down what the binary bytes should be. I note in the article under discussion, these are referred to as "Unique binary data" and "magic numbers" to help identify the origin of a PDF file, and are tabluated in the article (which table is reproduced in this report as figure 2.11. As can be seen from that table, in the case of OpenOffice/LibreOffice the magic number is (in hexadecimal) `c3 a4 c3 bc c3 b6 c3 9f` , whereas in TEX-based systems the magic number differs.

63. Inspecting the second line of the BWP PDF, as shown in figure 2.3, the binary bytes of that line have been interpreted as if they were text and displayed as `%Ã¤Ã¼Ã¶ÃŸ+` and so do not allow for direct comparison. However, viewing the same data in a hex

Unique Binary Data (in hexadecimal)	PDF producer tools	OS- Windows, Linux, Mac OS X	Distribution- TeXLive/MikTeX
0xE2E3CFD3	Acrobat Distiller	3 OSs	-
0xB5B5B5B5	Microsoft Office Word	Windows	-
0xD0D4C5D8	pdfTeX	3 OSs	TeXLive and MikTeX
0xD0D4C5D8	**LuaTeX**	**Linux**	**TexLive**
0xCCD5C1D4C5D8D0C4C6	**LuaTeX**	**Linux**	**MikTeX**
0xCCD5C1D4C5D8D0C4C6	**LuaTeX**	**Mac OS X and Windows**	**TeXLive and MikTeX**
0xE4F0EDF8	xdvipdfm	3 OSs	TeXLive and MikTeX
0xc7ec8fa2	GhostScript	3 OSs	TeXLive and MikTeX
0xc3a4c3bcc3b6c39f	LibreOffice	Linux	-
0xC4E5F2E5EBA7F3A0D0C4C6	Mac Os X	Mac OS X	-
0xB5EDAEFB	cairo	3 OSs	-
0xD3EBE9E1	Skia	3 OSs	-
0xF6E4FCDF	PdfLaTeX	online	-

Figure 2.11: Table 2 of the article "Robust PDF Files Forensics Using Coding Style" at Exhibit AR1

{H/324}

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	25	50	44	46	2d	31	2e	34	0a	25	c3	a4	c3	bc	c3	b6	%PDF-1.4.%Ã¤Ã¼Ã¶
00000010	c3	9f	0a	32	20	30	20	6f	62	6a	0a	3c	3c	2f	4c	65	Ã.2 0 obj.<</Le
00000020	6e	67	74	68	20	33	20	30	20	52	2f	46	69	6c	74	65	ngth 3 0 R/Filte
00000030	72	2f	46	6c	61	74	65	44	65	63	6f	64	65	3e	3e	0a	r/FlateDecode>>.
00000040	73	74	72	65	61	6d	0a	78	9c	ad	5c	4b	8b	24	b9	11	stream.xœ-\K‹$¹.

Figure 2.12: Magic Bytes from the BWP PDF, viewed in a Hex Editor

editor allows the hexadecimal encoding of those bytes to be inspected, as shown in 2.12, in which the relevant characters and their corresponding hexadecimal digits are highlighted in red. As can be seen, the interpreted 'text' **%Ã¤Ã¼Ã¶Ã** corresponds to the hexadecimal **c3 a4 c3 bc c3 b6 c3 9f**, which is to be expected of PDF documents created with OpenOffice.

64. By contrast, documents created in TEX engines would have different binary digits in the PDF header depending on which engine was used. The table at figure 2.13 shows the digits for LuaTEX. Inspecting the relevant part of the source code of LuaTEX[8], excerpted at figure 2.13, shows that the binary digits in that case are actually created in an arbitrary way, by adding 128 to the digits of "LUATEXPDF".

Conclusion on the Bitcoin White Paper

65. There are a number of differences between the PDF file of the original Bitcoin White Paper and the output of a standard LATEX installation, which I have explained above.

66. While most of the typographic differences (such as hyphenation and choice of fonts) may

[8] https://github.com/TeX-Live/texlive-source/blob/1164c633ef432434161638cd05a2a95d2837f47d/texk/web2c/luatexdir/pdf/pdfgen.c#L1007

```
/*tex Write \PDF\ header */
pdf_printf(pdf, "%%PDF-%d.%d\n", pdf->major_version,
pdf->minor_version);
/* Some binary crap. */
pdf_out(pdf, '%');
pdf_out(pdf, 'L' + 128);
pdf_out(pdf, 'U' + 128);
pdf_out(pdf, 'A' + 128);
pdf_out(pdf, 'T' + 128);
pdf_out(pdf, 'E' + 128);
pdf_out(pdf, 'X' + 128);
pdf_out(pdf, 'P' + 128);
pdf_out(pdf, 'D' + 128);
pdf_out(pdf, 'F' + 128);
pdf_out(pdf, '\n');
```

Figure 2.13: Excerpt of LuaTEX source code, showing how the header of a LuaTEX PDF is constructed

be explained by stylistic or other personal choices of the author of the White Paper, that is not true of the technical divergences such as the presence of the `/DocCheckSum` trailer element, or the binary digits in the header. For a TEX-based system to have produced the White Paper would have required the following extensive and subtle modifications:

a. modifying the basic code of a TEX engine to output predictable prefixes for subset fonts;

b. modifying the engine to include a `/DocChecksum` element in the trailer of the PDF file,

c. modifying the engine to change the binary digits output in the header of the file,

d. modifying the engine to encode fonts as TrueType fonts,

e. modifying the font used, to present kerning for every character pair, and

f. modifying the engine to output text encodings differently within the PDF.

67. While these modifications are theoretically possible (in that it is possible to modify the code of any open-source software), they seem like a lot of trouble for no discernible benefit; a far cry from the considerations that usually give rise to TEX extensions, such as XeTEX and LuaTEX, and indeed from the decision to use LATEX itself (which is designed to allow a separation of content from formatting concerns, allowing content to be written in plain text without concern over the format). The technical steps required are also rather advanced, as they would require knowing the internals of the TEX engine, the

TrueType font format, and PDF font format. It is to be wondered why anyone would do that. I will not speculate as to the latter but do point out that although my initial impression was of a typographic style close to LaTeX's, the deeper I dug into the original Bitcoin White Paper, the less convinced I became that any version of LaTeX could have produced that file. It would require that someone took great care to create a specialised TeX engine that had the capability to mimic the output of OpenOffice very closely while retaining the superficial appearance of a LaTeX document, yet not looking exactly like a real LaTeX document either, and not taking advantage of many of the core features of LaTeX. By far the more logical conclusion is that OpenOffice, instead, produced the document.

68. For all the reasons above, and after putting all the possibilities in balance, I consider it extremely unlikely that the Bitcoin white paper has been produced with LaTeX.

69. Finally, I previously noted that Bird & Bird also provided me with copies of two other versions of the Bitcoin White Paper, dated to October 2008 and November 2008 respectively. I have also looked at those versions in the same way as the March 2009 version. All three are very similar in each respect, and I have reached the same conclusion about those 2008 versions for the same reasons as the 2009 version.

Chapter 3

The LaTeX Files

70. On 22 December 2023, Bird & Bird sent me a folder called `TC` containing a number of files and instructed me that it had been indicated to them that some of these files, though not necessarily all, could be used to reproduce a copy of the original Bitcoin White Paper from them. I was asked to analyse them and form an opinion on which, if any, could be the source of the Bitcoin White Paper. I had also been made aware of, and agreed to abide by, the confidentiality terms under which the documents were provided.

71. I was also sent a file called `Compiled WP.pdf` which, I was informed, was Dr Wright's team's attempted compilation of a version of the Bitcoin White Paper from those provided LaTeX source files, alongside a document called "CSW8", which I understood from Bird & Bird was a witness statement from Dr Wright providing the details of the LaTeX environment he said he used to create the BWP. {E/23}

72. This chapter explains my analysis of the files, and I explain the steps I took and observations I made in chronological order.

73. I start by giving an overview of the content of the folder of documents provided, and then move on to an attempt to classify these files. This was a complex process, as there were many files and no guidance was provided to help me to understand the significance of the files.

74. I then move on to discuss the main body of the files, their diagrams, and my efforts to compile the various files using a setup that would match 2008. (In some cases, requiring me to fix their coding).

75. Finally, I explain the function and issues caused by the inclusion of various commands and packages which have allowed me to assess whether these files could have been used

in 2008-2009.

3.1 Overview of the contents of the folder `TC`

76. I found the structure of the folder a little confusing at first, but it can ultimately be summarised as:

    a. Font files, often in duplicate;

    b. Images in up to 3 formats, often including duplicates or near-duplicates within the same format;

    c. Knitr files[1] containing the White Paper's formulae and calculations;

    d. Fifteen (15) LaTeX files containing some purported versions of the Bitcoin White Paper and referring to some of the fonts and the images;

    e. One ancillary file used by LaTeX; and

    f. Other LaTeX files.

77. The full file listing is given in figure 3.1.

78. The font files are present in both the root folder and a sub-folder called `FontTT`, with a great deal of overlap: most, but not all, font files are found in both folders with the same name and their contents are identical byte-by-byte.

79. The image files are in PNG format in the main folder, but they are also in a sub-folder named `images` where they are present in PDF and TeX formats. In the latter case, the TeX files describing the images use a package called TikZ, on which I will expand later.

80. The ancillary LaTeX file mentioned above is the list of bibliographic references "`references.bib`" (in BibTeX format). There are some more LaTeX files, that I have not used to re-create the Bitcoin White Paper and have not looked into further as they do not seem to relate to the Bitcoin White Paper itself, and for lack of time.

81. The fifteen LaTeX files are the most interesting part: all but one of them contain prose identical or very similar to that of the Bitcoin White Paper, with varying amounts of LaTeX coding. An exception within the 15 is the file called `TimeC.tex` that contains a summary of the White Paper's text, but with different text relating to security considerations appended at the end. This file can thus not be the origin of the Bitcoin White

[1] Knitr files are files that combine a mix of LaTeX and the R programming language, which is a language used for statistics calculations. Although the R programming language has existed since 1993, Knitr was first released in January 2012 according to its announcements page at https://github.com/yihui/knitr/releases?page=6.

```
TC:
1_transactions.png           Code5.Rtex         TC9.tex
2_timestamp.png              E-Cash-main.tex    Tex002.tex
3_proof-of-work.png      }   ECash-Main01.tex   TimeC.tex
4_reclaiming-disk.png        FontTT/            Timecoin.tex
5_spv.png                    FormalProof.tex    TimesNewRomanPS-BoldMT.ttf
6_combining-splitting.png    IEEEtran.bst       TimesNewRomanPS-ItalicMT.ttf
7_privacy.png                Image1.tex         TimesNewRomanPSMT.ttf
BitCoin 2007.tex             Key-moves.tex      images/
BitcoinSN.tex                OpenSymbol.ttf     main.tex
CenturySchoolbook-Bold.ttf   P1.tex             main01.tex
CenturySchoolbook.ttf        PoW.tex            main02.tex
Code1.Rtex                   TC.tex             main03.tex
Code2.Rtex                   TC001.tex          references.bib
Code3.Rtex                   TC2.tex
Code4.Rtex                   TC8.tex

TC/FontTT:
ArialMT.ttf                  OpenSymbol.ttf                TimesNewRomanPSMT.ttf
CenturySchoolbook-Bold.ttf   TimesNewRomanPS BoldMT.ttf
CenturySchoolbook.ttf        TimesNewRomanPS-ItalicMT.ttf

TC/images:
Image1.pdf   Image2.pdf  Image3.tex  Image5.pdf      Image5_old.tex  Image7.pdf
Image1.tex   Image2.tex  Image4.pdf  Image5.tex      Image6.pdf      Image7.tex
Image1a.tex  Image3.pdf  Image4.tex  Image5Bold.pdf  Image6.tex      name.tex
```

Figure 3.1: File listing for the TC folder

BitCoin 2007	BitcoinSN	E-Cash-main	ECash-Main01	P1	TC	TC001
A	**B**	**C**	**D**	**E**	**F**	**G**
TC8	TC9	Tex002	Timecoin	main	main01	main02
H	**I**	**J**	**K**	**L**	**M**	**N**

Table 3.1: The fourteen LaTeX files (`.tex` extension omitted)

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**
1062	1005	718	694	251	346	466	576	576	640	461	1061	694	467

Table 3.2: Size of the fourteen files, in lines

Paper, even though the subject matter is similar; that leaves a total of fourteen (14) candidates which I have considered as possible LaTeX sources of the White Paper.

3.2 Attempt at a classification of the files

Labels for each of the fourteen candidates

82. Faced with this relative abundance of LaTeX files, I needed to adopt a more systematic approach. I chose to refer to every file according to the equivalence table 3.1, using capital-letter labels. Throughout this chapter I will generally refer to the files by their label as in the table 3.1.

Note on file sizes

83. One of the first things I have to remark upon is that the files had very different lengths (measured in lines of code), as shown in table 3.2. The shortest file is **E** with 251 lines, followed by **F** that has 346 lines, then all the way to 1062 lines for **A**. **E** is special in that it is heavily excerpted, containing only the first few paragraphs and the conclusion of the text of the White Paper. Other than that, the differences between the files are mainly due to the amount of LaTeX code, with very few differences in the textual content. I will say more on that latter point in the next section, and will for the time focus on the LaTeX code, which is almost exclusively concerned with altering the visual appearance of the document, changing fonts, metadata, or the dimensions of margins and other graphical elements. It may also use commands to change spacing within a line of text, or to place a figure at a fixed point on the page.

Starting point for classification: parameters of some common commands

84. As a starting point, I looked for other criteria to try and classify the files and chose three LaTeX commands that are invoked exactly once in almost all files: `\geometry`, `\begin{enumerate}`, and `\begin{adjustwidth}`, used respectively to set the dimensions of the page; start a numbered list; and reset the widths of the page the margin. They can each take different parameters, and it is the different values of these para-

Geometry package options:	
`\geometry{left=3cm,right=3cm,top=2.5cm,bottom=2.5cm}`	**F**
`\geometry{left=3.81cm,right=3.0cm,top=3.42cm,bottom=4.0cm}`	**E K N**
`\geometry{left=3.82cm,right=3.84cm,top=3.445cm,bottom=3.661cm}`	**A B C D G H I J L M**
adjustwidth options:	
`%\begin{adjustwidth}{2em}{0pt}`	**F** (commented out)
`\begin{adjustwidth}{13.5mm}{15.0mm}`	**C D M**
`\begin{Adjustwidth}{13.48mm}{14.81mm}`	**A B G H I J L**
`\begin{adjustwidth}{13.48mm}{16.81mm}`	**E K N**
enumerate options:	
`\begin{Enumerate}[itemsep=0.05pt,...,leftmargin=11.30mm]`	**A B L**
`\begin{enumerate}[itemsep=0.0pt,...,leftmargin=11.0mm]`	**C D M**
`\begin{enumerate}[1)]`	**F G H I J K N**

Table 3.3: Parameters to Geometry, Adjustwidth, and Enumerate in the 14 candidate TEX files

meters among the set of candidates that will help with the classification, as shown in table 3.3. It can be seen that all files are present in all tables, with the exception of **E**, missing from the last table. This is because it is much shorter and does not contain the enumeration from section 5 of the Bitcoin White Paper (or indeed any of that section).

85. Looking at the table 3.3, it is possible to draw some conclusions about groupings:

    a. We can see groupings of files that are always on the same row in each table: first **E K N**; then **C D M**; and finally **A B L**.

    b. We can also see that **F** seems to have a special status, in that it is in its own category for two criteria, and also has the simplest sets of parameters generally. We can reasonably assume that this file is the oldest in the genesis of all fourteen files, which is corroborated by it being the shortest one of the complete files[2]: it also has the simplest structure, containing only standard LATEX markup to define sections, enumerations, etc.

    c. Then, considering the geometry table, we can expand on that latter notion by pointing out that it is a reasonable assumption that parameters with very precise values, such as 3.445cm and 3.661cm on row 3 of the table 3.3, most likely came later than less precise values, such as 3.42cm and 4.0cm shown on row 2 of that part of the table. This does not mean that the files with the more precise values are necessarily older than the other ones, but rather that they can be considered as a separate branch, or 'lineage', in the 'genealogy' of the fourteen candidate files.

    d. This reasoning points to **E K N** being a separate branch, and I will refer to it at the **First Branch**.

[2]The exception being **E**, which contains only an extract and therefore is not a complete file.

`pdfproducer=OpenOffice.org 2.3,`	`pdfproducer=OpenOffice.org 2.4,`
`pdfstartview=FitH,`	`%pdfstartview=FitH,`
`pdfcreationdate=D:20061122010000,`	`pdfcreationdate=D:20090324173315,`
`pdfmoddate=D:20061122010000 % i`	`pdfmoddate=D:20090324173315 % i`
empty line	

Table 3.4: Comparison of **A** and **L**

e. Considering the next part of the table 3.3, `adjustwidth`, supports the view that **F** is the simplest file (with no active `adjustwidth` command at all, since it is commented out). The other rows however imply some different branching, as **C D M** have simpler sets of parameters to `\begin{adjustwidth}` than the other files. I think that a reasonable interpretation is that the indentation has been changed later in several branches separately, as it can be seen that from their size that **C D M** are among the more complex LaTeX files (but not the most complex ones). With that in mind, I will now call these three files the **Second Branch**, with all the remaining ones being a **Third Branch**.

f. Within that **Third Branch**, **A B L** have a special status, due to their size at over 1000 lines of code each.

86. Within these groupings, we can start comparing the files two-by-two to refine the classification. For example:

a. comparing **D** and **C** line-by-line shows that the latter contains all of the same lines as the former, with about 20 lines added: this points to the conclusion that **C** was almost certainly created from **D**, and is thus more recent.

b. Another example is the comparison of **A** and **L** as shown in table 3.4: the values of these lines, and the fact that one line is commented out in the latter, tends to indicate that **A** precedes **L** in the genealogy.

87. By using these techniques for all the remaining files, I came to the classification shown in figure 3.2.

88. I have not based any conclusions on this genealogy, as textual criticism is not my area of expertise. I have also not been asked to try to establish an editing order for the various files, so it was not necessary to draw any firmer conclusions. However, I have used it as a guide to help me navigate the different files and to better understand the differences and the commonalities. I thought it important to present it here because it was part of my instruction to set out how I arrived at my conclusions, not just what my conclusions were.

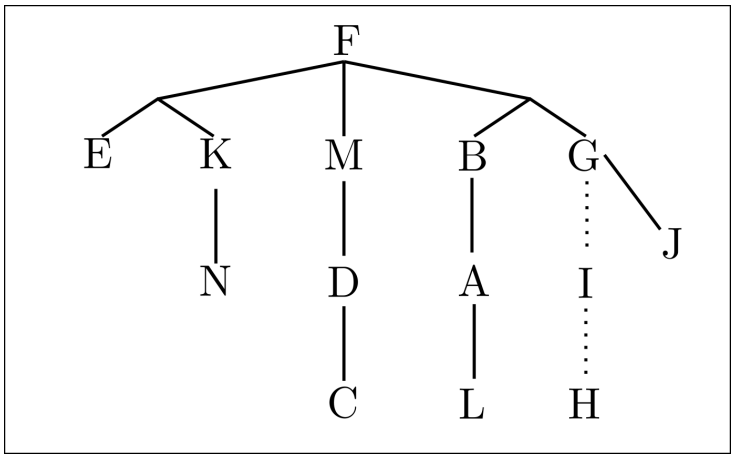


Figure 3.2: Genealogy of the fourteen candidates LaTeX files

3.3 Main text of the candidate documents

89. I extracted the text of all the fourteen files, without any code commands, and formatted them as plain text. The resulting files are Exhibited to this report as Exhibits AR6 to Exhibit AR19. This allowed for easy comparison between files, using the utility "`diff`", a data comparison tool that can compare two files line-by-line. Using the classification I had just made, I could thus look for relevant differences between all the files when it comes to their textual content. The differences I found were rather small: a few words are changed here and there, some typographical errors are corrected, and one phrase is replaced by another very occasionally. The largest number of differences is found by comparing **D** and **M**, as in table 3.5. Between these two files, the differences show up mostly in the Abstract section. The text of **D** is not identical to the October 2008 version of the BWP, as it has a title "Electronic Cash Without a Trusted Third Party": Bird & Bird have instructed me that this is identical to an earlier draft of the BWP's title and abstract which I have not seen, while the text of **M** is identical to the March 2009 version.

{H/329} - {H/342}

3.4 The diagrams

90. I now turn to the diagrams included in the different LaTeX files. The original Bicoin White Paper has seven small diagrams, and they are all present in each one of the fourteen files, with the exception of **E**.

D `ECash-Main01.tex`	**M** `main01.tex`
Electronic Cash Without a Trusted Third Party	Bitcoin: A Peer-to-Peer Electronic Cash System
the burdens of going through a financial institution	going through a financial institutions
offer	provide
trusted party	trusted third party
As long as honest nodes control the most CPU power on the network, they can generate	As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate
any attackers	attackers
intermediaries or third parties	third parties
Even small completely non-reversible transactions	Completely non-reversible transactions
the burdens of mediating disputes	mediating disputes
intermediary	third party
is valid	counts
do not	don't
the complete set of	all
honest nodes	majority of
longest honest chain	longest chain
they are	they're
broadcasted	broadcast
greedy or selfish	greedy
it is	it's
can not	can't
be alerted to	alerted
intermediary	third party
does not	doesn't
can not	can't
does not	doesn't
the most	a majority of

Table 3.5: A comparison of the textual differences between **D** and **M**

Different methods of embedding diagrams in the 14 candidates

91. Some of the LaTeX files include the diagrams as external files in PNG format that are present in the root `TC` folder. These can be seen in the file listing of section 3.1, under the heading '`TC`'. The command invoked to include these PNG files is `\includegraphics`, as in E:

    ```
    \includegraphics[width=0.75\linewidth]{1_transactions.png}
    ```

    This command above "includes" the graphics file called `1_transactions.png`.

92. In other of the LaTeX files, diagrams are included in mixed formats. Specifically, diagrams 2 and 7 are included as PNG files, but the first diagram is instead coded with LaTeX commands from the package TikZ (discussed further at section 3.7.5). These are implemented either by including that TikZ code in the main TeX file directly, or by putting it in a different file that is then incorporated by reference (and the two options make no difference to the LaTeX compiler).

93. In one instance, diagram 1 is simply missing.

94. Finally, in yet other of the candidate LaTeX files, the diagrams are included as PDF files (which can be seen within the listing at figure 3.1 within the folder `TC/images`. These PDF files, in turn, have been made using LaTeX TikZ. These are included in the main file with the same command as for the PNG files, `\includegraphics`.

95. When viewing these various images within a PDF document, the visual appearance is the same. However, each method results in quite a different encoding when viewed within the page stream of the final PDF file produced by LaTeX.

96. I also noted that the version of the PDF format used by the various PDF diagrams is PDF version 1.5, which is more recent than the version of the original BWP (which used 1.4). This latter version number is also set explicitly in several of the fourteen candidate LaTeX files as the PDF format to be output. It is not necessarily a problem that the version numbers don't match between the original BWP and the candidates, but it was contrary to my expectations of what I would expect when embedding one PDF file (an image) inside another (the main document). If a document is created as PDF version 1.4 but embeds an image made with the later PDF version 1.5, it may cause compatibility issues since the PDF viewer or interpreter would not expect to find v1.5-compatible features within a v1.4 file. If a file included in the main file were then to use features from a later version, the PDF interpreter might behave inconsistently.

Metadata specified in the LaTeX files

97. Interestingly, I saw that certain of the LaTeX files contained commands that would explicitly set the metadata of the output PDF files, to record a prescribed date and

```
% Configure hyperlink colors and styles for URLs
\usepackage[hidelinks]{hyperref}
%\usepackage{hyperref}
\hypersetup{
  colorlinks=true,
  linkcolor=black,
  citecolor=black,
  urlcolor=black,
  % other options
  pdftitle={Bitcoin},
  pdfauthor={Satoshi Nakamoto)},
  pdfproducer={OpenOffice.org 2.3},
  pdfcreator={Writer},
  pdfnewwindow=true,
  pdfsubject={Bitcoin: A Peer-to-Peer Electronic Cash System},
  pdfkeywords={Micropayments, Electronic Cash, Peer to Peer},
  pdfstartview={FitH},
  %pdffilemode={UseNone}
  pdfcreationdate={D:20080621170945},
  pdfmoddate={D:20080621170945} % i
  %pdffilemode={UseNone}
}
```

Figure 3.3: Command `\hypersetup` used in **C** to set the metadata

software irrespective of which software was actually used to compile them. The result is that some of the provided source documents would, even if compiled with a LaTeX engine today, nevertheless output a PDF which declared in its metadata that it was created in 2008 with a version of OpenOffice. Figure 3.3 shows an example of this command being used in **C**.

Differences in the figures

98. It can be seen that in the `Compiled WP.pdf` from Dr Wright's team, and also in the PDF versions of the image files, figure 4 of the BWP did not perfectly match in its content. Specifically, the label of the box "Hash0" in the BWP is not represented accurately in Dr Wright's version of that figure, which labels that box as "Hash01", as shown in figure 3.4.

99. Reviewing these PNG format diagrams, I noted that while diagrams 2 to 7 from that listing use Arial as a font (which matches the original Bitcoin White Paper), diagram 1 uses a different font that I have managed to identify as *Liberation Sans*, and so did not exactly match those of the Bitcoin White Paper itself.

100. Inspecting also the shape and placement of lines and arrow heads within Dr Wright's files, it is possible to observe subtle differences in shape and placement, as seen in figure

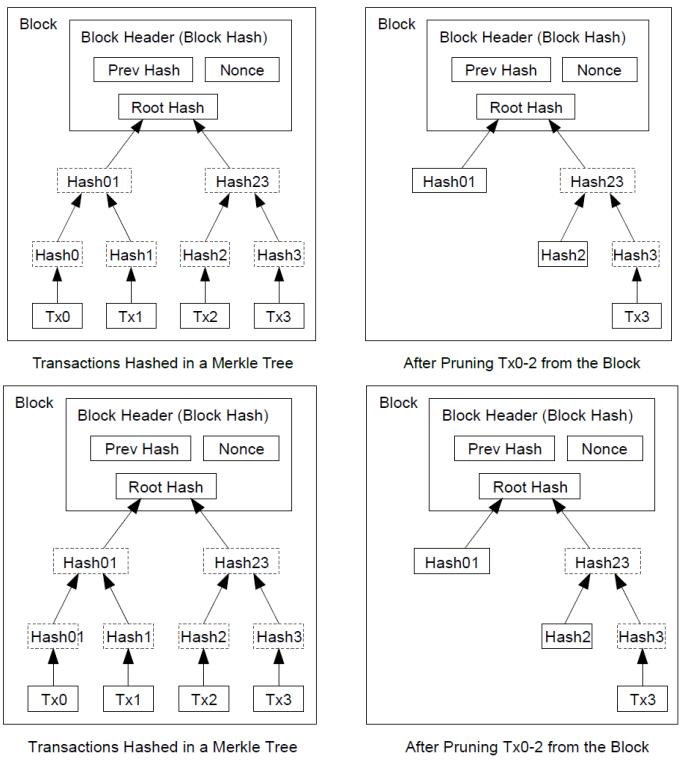


Figure 3.4: Comparison of the BWP's Figure 4 as it appears in the BWP (top) with figure 4 of Dr Wright's PDF files (bottom). Note the difference in label 'Hash01' toward the bottom left, which overflows the bounding box.

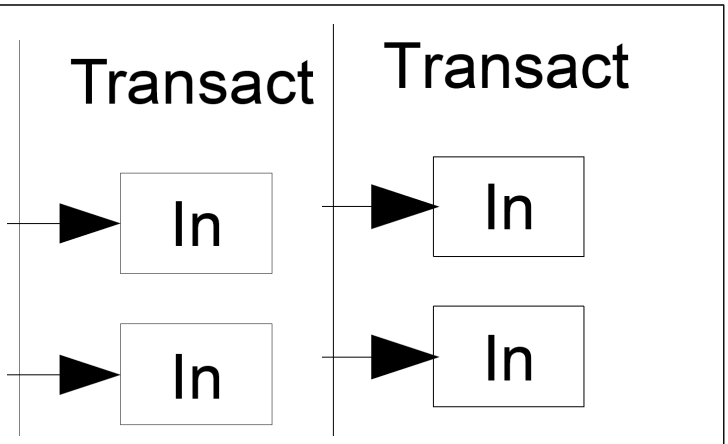


Figure 3.5: Magnified comparison of arrows as seen in the BWP's Figure 6 (left) with arrows seen in figure 6 of Dr Wright's PDF files (right). Note the subtle differences in shape and placement against the bounding box to which it points, as well as the differences in line thickness.

3.5. The arrowheads in the diagrams are important, and I will also return to these later.

3.5 Compiling the LaTeX Files

101. Having made all these observations, I needed, in order to gain a better understanding of the LaTeX code, to try and compile the fourteen files.

Using 2008-2009 LaTeX software

102. I first needed to create an environment that was as close as possible to what would have been used in late 2008/early 2009. This would allow me to assess the end result of each of the files, and compare them to the BWP.

103. I chose to use TeX Live, which is the distribution I am most familiar with and which has been published once a year since 1996[3]. In March 2009, the latest available version was TeX Live 2008, dating from 22 August of that year. It installed smoothly on a recent computer I'm using, running Linux (I tested the installation process on two different Linux distributions, Debian and Ubuntu).

104. That resulted in a TeX setup very similar to what a LaTeX user would have used in the second half of 2008 and first half of 2009. Compared with the 2023 edition, TeX

[3] Although I note that Dr Wright discusses the use of MiKTeX in his witness statement, that choice did not affect my analysis for reasons that are clear from the discussion which follows.

Live 2008 contained considerably fewer files but still has a lot of them, at over 70,000. Notably, LuaTeX had just been added to TeX Live that year, while XeTeX had been in the distribution since the year prior, 2007.

105. Even though Dr Wright states that he used LuaLaTeX, I elected to use XeLaTeX in order to compile each one of the fourteen files, for reasons I will explain at section 3.7.1.

Compiling the candidate files

106. I installed the fonts from the `TC` folder on my computer in a way that could be used by XeTeX and proceeded to compile the fourteen files one by one.

107. This was not as simple as it sounds, because all but one of the candidate files generated a number of errors connected with some specific packages or the commands therein. That is to say:

    a. It was not possible to compile any of Dr Wright's LaTeX files using 2008-2009 software, except one.

    b. The only file which did compile under my 2009 installation was **F**. That is also the file which is simplest in structure, and which does not include any commands which cause the output to resemble the formatting of the BWP.

108. I felt it necessary to compile as many of the files as possible, as best I could, if I was to analyse them properly. I resorted to various techniques to overcome the obstacles that were presented by the errors, adjusting the code to the minimum extent necessary to achieve a PDF output which could be compared to the BWP. The type of changes needed will also be clear from the later sections of this Report.

109. In this way I could then obtain a result for most, but not all, of the fourteen files:

{H/343} - {H/353}

    a. The eleven candidates that I could compile in this way are Exhibits AR20 to AR30.

    b. The files I could not compile at all under TeX Live 2008 were **A**, **C**, and **L**. I discuss the problems encountered with these packages in the next section. There is, of course, nothing to exhibit in that respect.

Errors and Warnings

110. It is important to understand what these errors mean: when LaTeX is run, it will by default stop and ask for user input as soon as it encounters an unknown package, command, or syntax error. This does not imply that every error will result in a visible difference in the document, but it can prevent the document from compiling at all without further input. What is common to all these causes of errors is that – with normal settings – LaTeX will stop during the compiling process, and require the user to do something before it can continue. This was referred to as "interactive" in the 1970s. It is possible,

and quite easy for the advanced user, to deactivate this interactive mode and simply ignore the errors, and this is indeed the default setting today in Overleaf: the user has many more ways to interact with the system today than it did almost fifty years ago. It is thus conceivable that a LaTeX user had in 2009 set up their environment to silently ignore errors, and I do therefore not attach too much importance to the presence of an error; but I do need to note it.

111. When a package is loaded, it can have many different effects, the most important ones being chiefly to define settings, or alter pre-defined defaults, or to create new commands. Most packages do both of those things, and if a LaTeX file tries to load a package that is absent, it will, in the former case, result in that the desired settings will not be set; in the latter, the desired commands will be undefined. If new commands are specified in a LaTeX document but the package that defines them is not loaded, those commands (if used) will not have the desired effect in the document.

112. In the course of compiling the fourteen files, I also noticed that a number of packages were emitting warnings, that would not have caused the compiler to stop, but were nonetheless signalling that something was amiss. Warnings are lesser errors which can often be safely ignored, but they may have effects on the emitted PDF. Time did not allow me to look into these less-problematic packages, and except for noting that microtype (used for finer typographic settings) was one of them. I will say no more on this issue.

113. In the next section, I give an overview of the problematic packages that have formed the focus of my analysis. I need to point out that all of the fourteen files use more packages than the ones I present in the next section; in some cases many more. This is entirely within normal practice and I could not possibly describe every single package used without inflating this report to an unreasonable size. There is also the issue of relevance: the reason why I single out a few packages is hopefully obvious, namely because of the errors they generate, while other packages are of little relevance and can be ignored for this purpose.

Matrix of the Problematic Packages

114. I give in table 3.6 a matrix of the fourteen files against all the packages emitting errors, and a few other significant factors. A key is found at table 3.7. The "OBWP" column gives the properties expected of a LaTeX file in order to have been the origin of the Bitcoin White Paper in March 2009: none of the packages from section 3.7 should be present in that case (since they post-date the relevant dates), and the other pieces of information I selected should have the expected values I noted.

Two files which stand out from the 14

115. Looking at the matrix, I was able to deduce which one of the fourteen files had been used by Dr Wright's team to create the file `Compiled WP.PDF`: it was the one where the

	OBWP	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**
`fontspec`	N	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y
`hidelinks`	N	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y
`unicode-math`	N	Y	Y	Y	Y	N	N	N	N	N	N	N	Y	Y	N
New `eso-pic`	N	Y	Y	Y	Y	N	N	N	N	N	N	N	Y	Y	N
`arrows.meta`	N	Y	Y	Y	Y	N	N	N	Y	Y	Y	N	Y	Y	N
`\newgeom`	N	Y	Y	N	N	N	N	N	Y	Y	Y	N	Y	N	N
`luacode`	N	Y	N	Y	N	N	N	N	N	N	N	N	Y	N	N
Excerpt	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N
Text	TNR	?	TNR	?	TNR	TNR	CM	TNR	TNR	TNR	TNR	TNR	?	TNR	TNR
Maths	TNR	?	CM	?	CM	N/A	CM	CM	CM	CM	CM	CM	?	CM	CM
Images	direct	PDF	PDF	PDF	PDF	PNG	PNG	PNG	mix	mix	mix	PNG	PDF	PDF	PNG*
Date	2009	2006	N/A	2008	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	2009	N/A	N/A
Producer	OOo 2.4	Ooo 2.3	N/A	OOo 2.3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	OOo 2.4	N/A	N/A
Author	SN	SN	SN	SN	SN	CW	CW	CW	CW	CW	CW	CW	SN	SN	CW

Table 3.6: Matrix of the fourteen files

information from the last three rows of the matrix 3.6 matches the values in the original Bitcoin White Paper, namely **L**.

116. The following day, I was informed by Bird & Bird that Dr Wright's counsel had nominated that file in addition, confirming my hypothesis.

117. At this point one other file stood out, which was **F**. That is the only one of the files which could have been created in 2009, since all the other ones use at least one package or option that did not exist at the relevant time. It can also be seen from the matrix and from inspecting **Exhibit AR23**, that **F** can not be the source of the original Bitcoin White Paper since none of the fonts match: where the BWP has Times New Roman as its main font for both the text body and the mathematical formulae, **F** uses LaTeX's default font, Computer Modern, throughout. Similarly, in the section beginning "Converting to C code" on page 9, that file uses the default monospaced font of LaTeX instead of Courier. An example of the difference can be seen in figure 3.6, which compares that part of the original BWP to the output of **F**.

{H/346}

3.6 Spacing

118. I also need to say a few words on spacing. The figure 2.1, it will be remembered, shows an example of overstretched horizontal spacing which I said was relatively rare in LaTeX because of hyphenation. If I had not had access to any LaTeX source file I would thus point to the lack of hyphenation as the reason for this somewhat inaesthetic typesetting; however, in some of the fourteen files, we can see that something different is at play: some spaces have been added deliberately.

119. The source code for that particular line in **L** is shown on figure 3.7, and should be easy to understand if I explain that the command `\;` adds a small horizontal space. The horizontal spacing, in that place, is thus due to a deliberate addition in the source code. This happens in many other places in **L** and a few others of the fourteen files, but not

Converting to C code...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

Converting to C code...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

Running some results, we can see the probability drop off exponentially with z.

Figure 3.6: Comparison of 'Converting to C Code' section of BWP (top) to the same section of Compiled Candidate **F** (bottom).

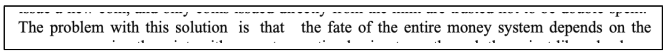
OBWP Original Bitcoin white paper
fontspec Loads fontspec (Y/N)
hidelinks Uses the `hidelinks` option from the package hyperref (Y/N)
New eso-pic Uses the command `\AddToShipoutPictureBG*` from the package eso-pic (Y/N)
unicode-math Loads unicode-math (Y/N)
arrows.meta Uses the TikZ library `arrows.meta` (Y/N)
\newgeom Uses the command `\newgeometry` from the package geometry (Y/N)
luacode Loads luacode (Y/N)
Excerpt The main body of the file is an excerpt (Y/N)
Text Main text font: Times New Roman (TNR), Computer Modern (CM), or ? when the document could not be compiled in TEX Live 2008
Maths Main maths font: Times New Roman (TNR), Computer Modern (CM), N/A when the document contains no formulae, or ? when it could not be compiled in TEX Live 2008
Images Format of the diagrams:
- **PNG** PNG format. The asterisk for **N** means that the first picture is missing.
- **PDF** The images have been made using TikZ in a standalone LATEX file that was compiled into PDF, then included in the main file
- **mix** All the pictures but the first are in PNG format, and the first one is coded in TikZ, either by inputting a file containing the TikZ code, or, with **J**, by writing the code direct into the main TEX source file
- **direct** The diagrams are coded direct using path constrution operators in the PDF's page content stream

Date Year of the date in the PDF metadata: 2006–2009, or N/A if not set explicitly. The year will thus show as e.g. 2023 or 2024 depending on when it was compiled.
Producer PDF Producer in the metadata: OpenOffice.org followed by a version number (OOo *n.p*), or N/A
Author Author in the body of the article: Satoshi Nakamoto (SN) or Dr Wright (CW)

Table 3.7: Key to the matrix

```
able \;to\; allocate many\; IPs.\;\; Proof-of-work\;is
\;essentially one CPU-one-vote.\;\; The majority
```

Figure 3.7: Spaces added explicitly in **L**




```
The problem with this solution \;is  \;that \; the fate of the entire money
system depends on the
```

Figure 3.8: Another overstretched line, and its source in **L**

all of them.

120. The result is sometimes really odd, as in figure 3.8 where I show the line extracted from the PDF file together with its source code. There are three different lengths for the interword space on that line: standard space; standard space plus `\;`, and two standard spaces plus `\;`. The spaces on either side of the word "is" are visibly wider, and even more so between "that" and "the". The reader will remember that **L** was the file designated by Dr Wright's as the source of their own attempt at reproducing the typesetting of the Bitcoin White Paper recently; and it can indeed be confirmed that it matches the visual appearance most closely, but in this instance it does not. The odd spacing is not present in the original.

3.7 Packages, options, and commands

121. In this section, I explain the various packages found in the candidate LaTeX files that have been the focus of my analysis.

3.7.1 Package fontspec and its use in conjunction with LuaTeX

122. The package fontspec is used to set custom fonts in LaTeXfiles. It is present in all but one of the candidate files (which is **F**). Fontspec provides settings and commands to specify the font, and in Dr Wright's LaTeX files it is used to set the same fonts which are also observed in the BWP.

123. The package fontspec was created in 2004 for the engine XeTeX, the first extension of TeX that could use almost any font. This was a very new feature that wasn't very well supported by the then-current font machinery of LaTeX, which is why fontspec was written, in order to offer a user-level interface to XeTeX's capabilities and to allow users to load these fonts into their LaTeXdocuments.

124. In March 2009, fontspec did not work at all with LuaTeX. At that time, any attempt to

run LuaTeX (or LuaLaTeX) on a file loading that package would result in an error, and the font-changing commands would be ignored. In the case of Dr Wright's LaTeX files, that implies that all the fonts would come out as the LaTeX defaults, Computer Modern (both for the text body and maths), as is indeed observed in Compiled version F (see {H/346} Exhibit AR23).

125. This was the reason for my choosing XeLaTeX rather than LuaLaTeXmentioned above: it was clear from the beginning that using the latter would have made it impossible for me to compile any of the fourteen files but **F** faithfully to their code. I do not know what engine Dr Wright's team used in order to produce the file `Compiled WP.pdf` that they provided, but since Dr Wright states he used LuaTeX in 2008/2009, I have presumed that that was also the engine they used recently.

126. Although Dr Wright states in his eighth witness statement that he used LuaLaTeX, this would not have been possible at the date of the Bitcoin White Paper, without a custom version of fontspec. I do not have any information about any custom environment, but this is the only logical way that the statement could be correct on this point, considering the situation at the time: The first version of the package to support LuaTeX was released in November 2009.

127. It would not have been easy to create such a custom environment. It may be helpful to give a short explanation of the technical issues: the reason why fontspec took a comparatively long time to be adapted to LuaTeX is because the underlying approach of LuaTeX was very different from that of XeTeXwhen it came to fonts. While XeTeX re-implements a number of processes of the TeX engines with the help of third-party code, LuaTeX approached things differently, and offered a number of ways to "hook into" the engine by replacing some parts with Lua code. By default, LuaTeX offered few additional features, but a lot of potential. That potential had already been tapped into by ConTeXt, a system that (like LaTeX) sits on top of TeX engines (and it constitutes what I called the "top layer" in section 1.3). At the turn of the year from 2008 to 2009, the most realistic prospect for using LuaTeX's theoretical capabilities in LaTeX was to take some of the Lua code from ConTeXt, and attempt to adapt it to work with LuaTeX. However, even that would not have been simple. As it happens, I wrote an explanation at the time (November 2008) which touched on the difficulties of porting fontspec to LuaTeX: See `https://tug.org/pipermail/xetex/2008-November/011213.html` for that longer {H/354} explanation from November 2008, which is also exhibited at **Exhibit AR31**.

128. A first step in that direction was the package luaotfload, which I described, together with other packages, in a talk I gave in July 2009 (under my previous surname Reutenauer). {H/355} I also wrote an article for the proceedings of the conference where that talk was held; the proceedings were published in October 2009: see **Exhibit AR32**[4].

[4] `https://tug.org/TUGboat/tb30-2/tb95reutenauer.pdf`

3.7.2 Package hyperref and its option `hidelinks`

129. The package hyperref defines commands to add hyperlinks and other information to the PDF file compiled by LaTeX. Iit is included in most of the fourteen files, with the option `hidelinks` as shown in the matrix.

130. While hyperref itself is older than 2008, that option `hidelinks` was only added to the package no earlier than 2010. The purpose of that option is that it "hides" the fact that links within the document are links, by displaying them without underlining, and without colour – whereas hyperref normally displays default links in their traditional style by underlining them and changing the colour to blue. It is a relatively innocuous difference in output, but at the point of compiling if an earlier version of the package were loaded with that option it will issue an error during the compilation. It also signals that these documents have either been created or modified in 2010, since to use that option before it was added to the package would imply guessing in advance what the option would be called. All of the fourteen files use that option, except for **F**.

131. It is possible to inspect the source code of releases of hyperref to determine a short period in which the `hidelinks` option was added, as follows:

    a. The changelog of hyperref is available at `https://github.com/latex3/hyperref/blob/6eeaaaa6919c75eef7476ae3ac447b86bb4a3a84/ChangeLog.txt#L731`. According to that changelog, the `hidelinks` option was added in version 6.82a dated 2011-02-05 (5 Februay 2011).

    b. The source code of the relevant file within version 6.82a can be inspected at `https://svn.gnu.org.ua/sources/hyperref/tags/hyperref-6.82a/hyperref.dtx`, which includes the code to specify `hidelinks`.

    c. The equivalent source code for the previous version, 6.81z, can be inspected at `https://svn.gnu.org.ua/sources/hyperref/tags/hyperref-6.81z/hyperref.dtx`. There is no code relating to `hidelinks`. I note that the date of that release is given in the changelog as 2010-12-16 (16 December 2012).

    d. This therefore indicates that `hidelinks` was released in February 2011. It was not available in the previous December 2010 version, narrowing the relevant period to under 3 months.

132. I do not exhibit the relevant source code (which would be hundreds of pages long) but it can be inspected at the URLs above, and the relevant excerpts are given in figure 3.9.

133. This package can also be used in order to alter the metadata stored in the PDF file, resulting in a file that may claim that it was created on a different date than it actually is, and to have been produced by a different computer program. This is an important point and I will come back to it.

```
2011-02-05  6.82a  Heiko Oberdiek
    * 6.82a
    * Options `hidelinks', `allcolors' and `allbordercolors' added.
    * \pdfstringdef: \begin, \end, \foreignlanguage catched.
    * Options `pdfpagelabels' and `pageanchor': some support
      for class with its page counters (slide/overlay/note/page) added.
    * Option `pageanchor': Double string escaping removed
      for default setting hypertexnames=true, plainpages=false.
```

```
\DeclareVoidOption{hidelinks}{%
  \Hy@colorlinksfalse
  \Hy@ocgcolorlinksfalse
  \Hy@frenchlinksfalse
  \def\Hy@colorlink##1{\begingroup}%
  \def\Hy@endcolorlink{\endgroup}%
  \def\@pdfborder{0 0 0}%
  \let\@pdfborderstyle\ltx@empty
```

Figure 3.9: Changelog (top) and source code (bottom) relating to the introduction of `hidelinks` into the hyperref package in December 2010 – February 2011.

3.7.3 Package unicode-math

134. The author of fontspec also developed a sort of companion package, for maths fonts - called unicode-math. In 2009, it was in its infancy supporting very few fonts and was not yet uploaded to CTAN: the initial release was on 3 June 2010[5].

135. The package was available before that on the code sharing platform GitHub. However, notably, Times New Roman, used for the formulae in the original Bitcoin White Paper, was not supported at that time.

136. Another fact to note is that the early versions of unicode-math suffered from the load-order problem explained at section 1.3: when used together with the amssymb package that defines additional mathematical symbols, the former needed to be loaded before the latter, i.e. it needed to be listed first in the source document. As it happens, all of the fourteen files that do load unicode-math do it *after* amssymb, meaning that TEX would have issued an error for every one of the 2307 mathematical symbols defined by the former package (in its early version of 3 August 2008).

137. It is theoretically possible that these issues could have been overcome by working on the source code to develop it further, privately. However, I note that in the fourteen source files it was, in the end, used for one single thing: the Greek letter λ. The package could in principle have been used to affect the font of all the letters and symbols of the equations, but my attempts at compiling the fourteen files indicate that it wasn't: none

[5] See `https://ctan.org/ctan-ann/id/mailman.1620.1275576440.2324.ctan-ann@dante.de`

of resulting PDF files used a maths font matching the BWP's, Times New Roman. I circumvented the issue outlined in the paragraph above by writing a one-line package to simulate the behaviour of how λ is mapped, and was able to proceed.

138. In summary, all that was necessary to simulate its behaviour was a single line of code, to map the λ to a glyph in a font; the package would otherwise have had no effect on the fourteen files. I have therefore not used the presence or absence of unicode-math in a LaTeX file as a criterion for my analysis. However, for reference, the files that do load the package are **A** to **D**, **L**, and **M**.

3.7.4 Package eso-pic and its command \AddToShipoutPictureBG*

139. Many files use the package eso-pic for placing pictures at specific coordinates on the page. That package already existed in 2009, and the relevant command was called `\AddToShipoutPicture*`.

140. However Dr Wright's files use that command under the slightly different name of `\AddToShipoutPictureBG*` (with 'BG' appended before the asterisk). The name of this command was changed in June 2010 [6]. If used before that, an error would be issued and the picture wouldn't be placed on the page at all. The name change, by the way, was made to better reflect the command's function, since it places a picture in the background of the page, like a watermark (a corresponding `\AddToShipoutPictureFG*` was added at the same time).

141. Because the new name was introduced in 2010, any file that uses that name must have been created or modified in or after that year: as with section3.7.2, it is not unreasonable to suppose that one could have guessed the exact new name before it was introduced. It is still theoretically possible for a file to have existed before the name change, with the command's old name, to then be modified to use the new name. Obviously, this can not be verified.

142. Those of the fourteen files that use the command under the new name are the same that load unicode-math: **A** to **D**, and **L** and **M**.

3.7.5 Package TikZ and its library arrows.meta

143. TikZ is a very big package for creating graphics in LaTeX. Using it, pictures can be defined programmatically with code that describes lines, curves, different geometric figures, etc. It is a very involved piece of programming, and is therefore broken down into many different "libraries" – essentially each one is a separate package with additional functionalities and features.

{H/356}

[6] See Exhibit AR33, Announcement on CTAN: `https://ctan.org/ctan-ann/id/mailman.2092.1276003629.2324.ctan-ann@dante.de`

144. One such library is arrows.meta, used in several of Dr Wright's files. This library defines additional shapes for arrowheads, such as a solid triangle. This is significant because the original Bitcoin White Paper uses solid triangles for arrowheads, whereas the TikZ default, which is in force when arrows.meta isn't used, looks different, like this: →.

{H/357}

145. The arrows.meta library was only released in September 2013: see **Exhibit AR34**, from `https://github.com/pgf-tikz/pgf/commit/a30f8b3f8dc285980c20e1638b9b25c4d00efe8d`, for the commit that introduced it. A file that uses the arrows.meta TikZ library would, when compiled before 2013, issue an error when the package is loaded, and the arrows will have heads as above, shown magnified on the right-hand side of figure 3.5.

146. Any file that loads the arrows.meta library could therefore not have been created in March 2009. There is however an additional difficulty, in that in several files where TikZ is loaded, it is not used at all, because the diagrams are included as PDF files that are compiled separately, and no TikZ commands are used in the main document. Those files are, again, **A** to **D**, and **L** and **M**. Those that do use TikZ commands are **H** to **J**.

147. Unlike other packages whose code could be easily replicated (like the issue with amsmath discussed above), I do not consider that is the same for TikZ. The complexity of TikZ cannot be underestimated: the main manual alone was 560 pages long in TEX Live in 2008; in TEX Live 2023, it is 1321 pages.

3.7.6 Package geometry and its command \newgeometry

148. While all of the fourteen files use the package geometry and its command `\geometry` to set the main geometry of the page (width, height, margins, etc.), some (**A**, **B**, **H** to **J**, and **L**) use the newer command `\newgeometry` to change the geometry mid-document.

149. This `\newgeometry` command only appeared in early 2010; the exact date was between 13 and 28 February 2010[7]). Before that, the command didn't exist and calls to `\newgeometry` would result in an error: the dimensions of the page would remain unchanged, looking quite unlike the BWP. Therefore, the files mentioned above could not have been used to create the BWP in 2009.

3.7.7 Package luacode

150. The package luacode was created in November 2010[8] for the LuaTEX engine and it defines a few convenience functions to make it easier to use the Lua language from within LuaTEX: when using TEX, some characters have special meanings and luacode deactivates these special behaviours so that these characters can be "seen" by Lua.

[7] see `https://ctan.org/ctan-ann/id/mailman.2919.1266100191.20360.ctan-ann@dante.de` and `https://ctan.org/ctan-ann/id/mailman.4502.1267396362.20360.ctan-ann@dante.de`, Exhibits AR35 and AR36

{H/358} - {H/359}

[8] See the announcement at `https://ctan.org/ctan-ann/id/mailman.532.1289233225.2307.ctan-ann@dante.de`.

151. It is a rather small piece of code and it was already possible to use Lua in LuaTeX (indeed, that was how the LuaTeX project started). The significance of this particular package is first of all its date; and the fact that it signals any LaTeX source file that includes it as being meant for LuaTeX. It is otherwise not always easy, and sometimes impossible, to identify which TeX engine a particular `.tex` file is supposed to be used with.

152. Only three files use luacode: **A**, **C**, and **L**. These could not have been in existence in 2009 and I was not able to compile them at all under TeX Live 2008, since they also use fontspec, which as I explained at 3.7.1 did not support LuaTeX then. I note that **L** was confirmed by Dr Wright's team as the source of the file `Compiled WP.pdf` they provided, but it can't in my opinion be the source of the original Bitcoin White Paper.

3.8 Maths font

153. This very short section is concerned with a simple, but salient fact: when compiled in TeX Live 2008, none of the fourteen files used the correct font for the formulae. Most often, it was the default LaTeX font of Computer Modern; in a few cases it was Cambria Math, the default maths font for unicode-math at the time; and in one file all the mathematical symbols came out as little blank boxes. The original Bitcoin White Paper uses Times New Roman in all its formulae.

154. On this basis alone, it is possible to rule out all fourteen files as a possible source for the White Paper, without use of a custom version of the unicode-math package that would have enabled Dr Wright to change the maths fonts to Times New Roman. The package was not present in either TeX Live or MiKTeX at the time, so that even if Dr Wright used primarily the latter, as we will see below, that cannot be a determining factor.

155. I have not seen any indication that such a package was on hand or used, but I discuss Dr Wright's description of his environment below.

3.9 Kerning in Times New Roman font

156. Having considered the fonts used in the various documents and the interplay with fontspec, I was then also able to complement one point I made in the previous chapter, about the kerning properties in the fonts. I said it was extremely unlikely that a font would have kerning for every single pair of glyphs, which is what would be required to output the encoding observed in the BWP PDF itself. I can now verify that fact: on inspecting the font, Times New Roman does indeed not have that many kerning pairs that would explain what I observed in 2.2.2.

157. On the contrary, figure 3.10 shows an extract from **G**'s page content stream where longer

```
/F2 9.963 Tf 138.07 -35 .54 Td[<002600550044004c004a>-249
<0036>-250<003a0055004c004a004b0057>]TJ -23.68 -11.96Td[
<005a0055004c004a004b005700110046
00230055004c0047004a00480056001000480056005700440057004800110052005500 4a>
]TJ 9.02 -11.95 Td[<005a005a005a00110055004c004700
4a00480056001000480056005700440057004800110052005500 4a>]TJ ET 0 G 0 g 0 G
0 g 0 G 0 g 0 G 0 g BT /F3 8.966 Tf 74.49 -141.66
Td[<00240045005600570055004400460057>]TJ
```

Figure 3.10: Extract of the page content stream for **G**

strings can be observed, similiar to those expected of a LaTeX-created PDF document (as seen in 2.6), but notably different to those observed in the BWP (see figure 2.5).

3.10 Conclusion on problematic packages, and OpenOffice

158. I have explained above various problematic packages and options which could not have been used in 2009 in the form they have in the candidate files. It is worth emphasising one point in particular, which is that those packages and options are the very functionality which causes the resulting PDF to be output with an appearance that matches the Bitcoin White Paper's in its fonts, geometry, spacing and the other aspects of the document which I have described above (at least in 2023). If any of those packages and options were not present, a PDF output from the files in the `TC` folder would look less and less like the Bitcoin White Paper for each of them which was not available.

159. In my opinion, Dr Wright's LaTeX source files would not have been capable of producing the White Paper in 2008/2009.

160. It is also worth pointing out that all of these observed characteristics are supported natively and by default in OpenOffice, and were in 2008-2009, which also coincides with my findings in Section 2.2.2 regarding the encoding of the BWP, as well as the metadata shown on the file itself.

3.11 Dr Wright's statement about his TeX environment

161. In his eighth witness statement, Dr Wright gives a lot of information about his computing and LaTeX environment in 2008/2009. I looked into it in order to better understand how (and whether) the Bitcoin White Paper could have been made in LaTeX then.

3.11.1 Operating system and TEX distribution

162. Dr Wright states that he used a mixture of Windows in Linux:

> "3. Whilst I was at BDO between 2005 and 2008, I built a custom environment for my own use at BDO, which used a Red Hat Linux operating system with VMWare and which allow him to use Windows XP from a virtual image (and a snapshot in time of the computer environment). I used Xen and Citrix (which allow users to have remote desktops and applications) to access that Windows XP and other desktop and server applications."

[...]

> **Linux**
>
> 163. 74. My primary Linux environment was integrated with Windows and supported Wine. This allowed me to run the environment across both platforms. I was able to access and work and my documents including those in LaTeX from both Red Hat Linux and Windows."

164. In my view these paragraphs are however somewhat inconsistent with one another: in para 3 Dr Wright writes that used Linux as the primary ("host") environment, and that Windows was installed as a virtual machine on top of it, but in para 74 it's the other way around, with Linux as host and Windows running in the Wine emulator. I do not know how to resolve that contradiction.

165. I also found the next paragraph somewhat puzzling:

> "75. MiKTEX was configured on Linux to use LATEX packages and compilers including:
>
> a. TEX Live: I used this as a TEX distribution as an alternative to MiKTEX on Linux.
>
> b. LATEX packages: packages included amsmath, geometry, graphicx, hyperref, babel, and fontenc as well as those listed before.
>
> c. Compilers: pdflatex for direct PDF generation, latex for DVI output, and bibtex for bibliography management."

166. The sentence arising out of a. is hard to interpret, because it seems to say at face value that Dr Wright "mixed and matched" packages from MiKTEX and TEX Live packages, which is not something that is envisioned by either of these distributions, and I have never heard of anyone doing so. This paragraph also contains the implication that in 2008/2009 Dr Wright uses MiKTEX on Linux, which is not possible since that distribution only started supporting Linux in 2018: See **Exhibit AR37**, from `https://askubuntu.com/`

{H/360}

`questions/888225/installation-of-miktex`, for but one online discussion where it transpires that MiKTEX was not available on Linux in February 2017, but had just recently started supporting Linux in February 2018. I have not been able to pinpoint a more precise date for its eventual release, for lack of time. I have considered whether this might be an error in phrasing, but one cannot be mistaken that Dr Wright really means Linux, since he uses the word twice in that sentence, and the section title just above paragraph 75 is just "Linux", and paragraph 76 indicates that he means Linux in contrast to Windows ("These tools offered functionality similar to what MiKTEX provided on Windows"). I have tried to think of possible explanations, but I do not know how that contradiction could be resolved.

167. I did nevertheless notice an interesting detail in that same paragraph, namely the mention of pdfTEX in c., which I designated at section 2.2.2 as the most likely TEX engine to have been used to produce the original Bitcoin white paper, if any TEX engine had been used at all. However, unfortunately all fourteen candidate files but **F** would fail to compile under pdfTEX since they use fontspec (which never supported pdfTEX), and the font of the document would therefore default to the Computer Modern font (as is {H/346} seen in Candidate **F**, Exhibit AR23 instead of Times New Roman, Century Schoolbook and Courier (as it is in the BWP).

3.11.2 Schedule of updates

168. Dr Wright writes, and he's right, that while LuaTEX was still being developed it could be unstable. However, his description of how he dealt with updates is difficult to reconcile:

169. "35. Given that I did not maintain a detailed list of versions for applications and packages, and only updated versions when necessary to avoid problems in running my computer, there was an inherent unpredictability in how LuaLATEX updates would interact with existing LATEX files. This lack of version control could lead to situations where a LATEX document that compiled correctly under one version of LuaLATEX might exhibit issues or behave differently under a patched version."

170. I cannot tell from the first sentence what the cause and the consequence is intended to be. It may mean that updates were made in order to resolve problems, but it also seems to mean that problems were caused by updating. It is of course entirely possible, with a program under active development, to run into some issues, update in the hope that it was fixed in a later version, to then be confronted with a new, unexpected issue. But to describe such events as a regular occurrence is in my opinion a stretch: during the early phase of development, when bugs are encountered, they are reported to the development team, who then strives to fix them as soon as possible and then publishes a new version with a brief description of what has been done along with a "change log" such that users following the development may update if they wish. This does not interfere with the

major new releases, because common practice is to fix all bugs, when possible, before adding new features. It was certainly how LuaTeX was being developed during the most active period. Even then, that most active period was, as I would describe it, already mostly over by the end of 2008.

171. I should also note that during the active development phase of LuaTeX, bugs did not usually result in subtle changes in the typeset output but rather in a crash of the whole application. I therefore do not fully agree with Dr Wright's account, at his paragraph 34, that changes to LuaLaTeX could cause changes in how documents were rendered, from subtle differences to more significant alterations in how document layout was handled.

3.11.3 Edition and compilation

172. On the subject of editing and compiling a LaTeX file, Dr Wright states the following:

> "**Saving vs. Compiling**
>
> 23. In LaTeX systems, the concepts of "saving" and "compiling" represent two distinct steps in the process of creating a document:
>
> 24. Saving:
>
> a. Saving refers to the act of storing the current state of your LaTeX document as a file on your computer. This is similar to saving a file in any other text editor or word processor. When you save a LaTeX document, you are essentially writing the text LaTeX commands, and any markup you have added, to a file with a `.tex` extension. This file does not yet contain any formatted output; it is simply the raw code and text that has been written.
>
> 28. In order to save files from MiKTeX I used TeX4ht, and PDF2LaTeX."

173. This did not make sense to me. Saving a file simply means writing it to the hard drive, as Dr Wright explains. The program TeX4ht, however, converts LaTeX files to HTML, a very different format. If one runs TeX4ht on a LaTeX file and saves the result, the LaTeX file itself won't be saved, it will be lost and an equivalent HTML document will be saved instead. I have not been able to identify what PDF2LaTeX is, save that there does appear to be a software project by that name hosted on GitHub which is dated from 7 years ago (a conversion program which indicates that it intended to convert a PDF to LaTeXsource by training a neural network: see `https://github.com/safnuk/pdf2latex`). I have considered whether this might just be a typo for pdfLaTeX, but do not think that would make sense either. I find it hard to understand what Dr Wright describes here, although it is possible that parts of the workflow described could be used for converting LaTeX documents into web pages for publishing online, and it does not seem to fit within

the context of saving and compiling (which is being discussed in that section of the statement).

3.11.4 No `.log` or `.aux` files provided

174. The last item I would like to discuss is the absence of `.log` and `.aux` files in the `TC` folder, the one containing the fourteen LaTeX files. Since these files are generated at the point of compiling any LaTeXdocument, I would usually expect to find one such pair of files for each LaTeX file that was compiled, which would likely provide helpful information such as about the tools used and when they were compiled. However, there were none, as can been seen in the full listing at Figure 3.1. I had Bird & Bird ask Dr Wright's counsel about them, and received the following reply:

> 175. "5. Finally, our client has not provided any ".log" or ".aux" files from Overleaf as he is not aware of such a feature or existence of these files, which as you explain, can contain the relevant metadata associated with the LaTeX files. This may have been a more recent feature or one that was never activated as he was not aware of it."

176. This paragraph is also very difficult to reconcile. The generation of `.log` and `.aux` files during compilation has existed since the very beginnings of LaTeX, as anyone running it on their computer can verify since they are very visible in the same directory where any files are saved. They can also quite easily be found when using LaTeX on Overleaf, although they are not prominently displayed (in my view they are quite accessible and to imply that they are hidden would be unfair to Overleaf's developers). The publicly-available archive of the backup tapes of Stanford Artificial Intelligence Laboratory (`https://saildart.org`), where TeX was created, contains LaTeX files from the 1980s where one can see the matching `.log` and `.aux` files.

3.11.5 Summary on the computing environment

177. In my opinion there are a number of inconsistencies in Dr Wright's description of his TeX environment from the time the Bitcoin White Paper was created, and there are also inconsistencies between the software which is described, as compared to the requirements for compiling the files in question in 2008-2009. I also understand that there was not an attempt to re-create that environment recently, although installation of a 2008 distribution of LaTeX is quite quick and taks only a few minutes. Although I note that Dr Wright mentions Overleaf, that service did not exist before 2012-2013 and cannot have been used in 2009 to create the BWP either.

3.12 Overall Conclusion

178. I am now in a position to summarise all my findings, and my overall opinion on the questions that I was asked to consider.

Considering the LaTeX files on their own merit

179. The background to my opinion is my analysis of the characteristics of the Bitcoin White Paper. Although I formed the opinion that the BWP itself was written in OpenOffice, I set that aside while considering the LaTeX files on their own merit. That analysis has, however, informed me as to what is to be expected of any suitable TeX-based source.

Diagrams

180. First, I consider the format and encoding of the diagrams used in Dr wright's documents to be important. I explained at section 3.4 that none of the fourteen files included diagrams in a way that matched precisely what I observed in the original Bitcoin White Paper, namely "direct" coding using PDF path construction operators, to create vector graphics within the resulting PDF. In all of the candidate LaTeX files provided, at least six of the seven diagrams are included as separate PDF objects (or as PNGs). This leads to important differences in how the resulting PDF objects are encoded. The reader will remember from section 3.7.7 that three files (**A**, **C**, **L**) would not compile at all under TeX Live 2008, but this has no bearing here since the format of the diagrams is written into the LaTeX file, as the parameter to the command `\includegraphics`.

181. There are also subtle but important differences between some of the images observed in Dr Wright's files, and those of the BWP (as I have explained at section 3.4 under the heading 'Differences in the figures'.

Fonts and Fontspec

182. Secondly, I consider the fonts. In my view none of the fourteen files can be reasonably expected to reproduce the font of the mathematical formulae of the BWP, namely Times New Roman. For all the eleven files that could be compiled at all, I could observe that when compiled in TeX Live 2008, the maths font was set as Computer Modern Maths. This means that the visual appearance of the BWP has not been reproduced for any of these eleven files, using 2008-2009 software.

183. Of the problematic packages I identified and described at 3.7, fontspec represents the most complex issue. It will be remembered that it did not support LuaTeX in March 2009. There is no great difficulty in imagining that XeTeX may have been used instead, but Dr Wright writes explicitly in his eighth witness statement that he used LuaTeX and only mentions XeTeX twice, in passing, in the whole document. This would then imply that he had a way of using fontspec with LuaTeX, since that package is loaded

by all but one of the fourteen candidate files. However, I do not think this is plausible because adapting fontspec to LuaTeX would have been a large and complex task and there is no indication that this was done.

184. I have also considered whether the LaTeX files I have been shown were created earlier, and then edited later to add fontspec. In my view that is not a plausible explanation, since it would leave unexplained how the document's fonts were set before. I explained at section 2.2.2 that pdfTeX could be used together with TrueType fonts to set the fonts, but Dr Wright states explicitly that this was not used:

185. "73. When I was exploring the capabilities of LaTeX, particularly XeLaTeX and LuaLaTeX, I came across the fontspec package, which had been available since 2008. However, in its early versions, it had far fewer features compared to what it offers now. Developers like Will Robertson and Khaled Hosny were actively contributing at that time, posting a variety of scripts, updates, and package extensions. These contributions were crucial for interfacing with OpenType fonts and ensuring broader compatibility within the LaTeX environment. Unfortunately, I do not have a comprehensive list of these extensions available."

186. The reason I quoted the paragraph is because Dr Wright's explanation states that he used fontspec for switching fonts in a document, which excludes the use of pdfTeX or any earlier engine.

The least likely candidates

187. Thirdly, I also consider the three files that I could not compile with TeX Live 2008 separately — namely **A**, **C**, and **L**. As shown in the matrix 3.6, this will show that these are the least likely to have been in existence in March 2009, loading as they do almost all of the problematic packages and options: **A** and **L** actually do load all of them, while **C** loads all but one. In particular, they all load luacode, which means as I explained at section 3.7.7 we can be certain that they were created on or after November 2010.

188. While I note that **L** was identified by Dr Wright's counsel as the source of their compiled version, as I have indicated, I consider that to be the least likely to be compatible with any form of LaTeX in 2008-2009.

Overcoming difficulties with the problematic packages

189. More generally, I have considered very carefully whether Dr Wright could have overcome the various issues listed above with the use, in 2009, of private versions of packages that would come to be released or updated later. Although I consider that to do so would be possible in theory (at least in some cases), I do not think this is a possible explanation in the present case because:

a. Making such packages could be a complex task, and I have not seen any indication that such complex packages were used and it is not described in Dr Wright's statement;

b. To the contrary, the syntax used in all of them perfectly matches for the versions released much later;

c. If Dr Wright had made his own packages, it would be surprising if he had used the identical syntax and function names to the versions which were eventually released publicly by their authors (in some cases many years later). While such an artefact may be a simple coincidence in one case, in my opinion it could not be an explanation across all of the many issues that have been observed;

d. I have in mind that different packages are different, and some are easier to modify than others. For example, the command `\AddToShipoutPictureBG*` in the package eso-pic is a relatively minor variation, since the command already existed under a different name, and would therefore have been a very minor modification. However, I cannot see any reason that the modification to command names in this way would be desirable when using a publicly-available and widely-used package, as it would have no practical effect on the document. Even if the modification to the command name was made, it is not clear that the name chosen would have precisely matched the name that was chosen by the package's developer when the future release was created;

e. In other cases, the packages are comparatively less important to the output of the files - for example, the `hidelinks` option of the package hyperref did not exist in 2009 and removing that would not cause a large change to the output, though it is still an observable difference in the way links would be displayed in PDF file;

f. However, even these less significant observations become more significant when considering the fact that they overlap with each other, and with other problems that I have observed; and

g. Finally, other packages are much more complex - as with fontspec above.

190. TEX is a programming language, and a very versatile one at that, such that one can theoretically do anything with it, including redefining all commands and packages to have a different meaning than the standard one. However, even under this hypothesis, I must conclude that none of the LATEX files could have been the source of the original Bitcoin White Paper in March 2009, nor have any of them been modified from a slightly different file that had been used at that time to create the BWP.

Taking into account both streams of analysis

191. Pausing to bring together the two parts of my analysis, I have approached this by two methods of reasoning that independently lead me to the same conclusion. The BWP was created in OpenOffice, as is evident from inspecting the relevant files at every level, from the fine details of its typographical presentation, down to the binary digits of the PDF. Similarly, the LaTeX source files provided cannot be the source of such a document, for the reasons given.

3.12.1 Observation about image creation

192. I will conclude this chapter with a hypothesis I formulated when looking back at the matrix 3.6, which was after I had formed the conclusions above, while trying to understand whether it was possible to ascertain anything more about how the candidate files were produced. I noticed that among those candidates that use fewer of those problematic packages, it can be seen that the diagrams are usually in PNG format. This led me to consider that the diagrams in the simpler LaTeX files had first been created as PNG files, that were then replaced by TikZ code over the course of editing (either via inclusion of a separate PDF file, or directly), and that brings me to the fourth and last chapter of this report.

Chapter 4

Diagramming and conversion tools

193. This chapter will be rather short because of time constraints towards the end of my instruction. During the conclusions above, I mentioned to Bird & Bird that automatic conversion tools existed which can be used to create LaTeX documents, one of which had been mentioned by Dr Wright in his statement. Bird & Bird asked me to comment on those tools and consider how they would convert the Bitcoin White Paper, and I have looked at three of them: conversion to LaTeXwith the extension Writer2LaTeX, the conversion tool called Aspose, and Pandoc.

4.1 OpenOffice

4.1.1 Writer2LaTeX

194. This tool is mentioned by Dr Wright in his eighth statement. It is an extension of OpenOffice that exports a document from OpenOffice format to LaTeX. The open-source repository contains a number of old versions and I chose two of them to review, 0.5.0.2 from 2 September 2008 and 1.9.9 from 16 June 2023.

195. I observed that Writer2LaTeX produces LaTeX code that is very concise, with minimal formatting commands. These are overall quite unlike those of Dr Wright's fourteen files that imitate more closely the original Bitcoin White Paper. In particular, the package hyperref is not used to alter the metadata in the output by Writer2LaTeX, although it is used in other ways.

4.2 Aspose

196. Aspose[1] is an online tool that converts PDF file to LaTeX. Converting the Bitcoin White Paper with Aspose produced an interesting result, where all the letters of the text were placed individually on the page; the converter specifies the exact coordinates for each character, leading to a very long and verbose file (with tens of characters of code needed for every letter that it typeset on the page). This is obviously not very useful in practice, and I ignored it.

197. The images, however, were more relevant, as they were encoded using TikZ. I observed that these matched exactly the TeX source files in the `TC` folder provided by Dr Wright. The output of Aspose images was similarly verbose and long to its text, and not simple to create by hand. However, I could recognise exactly the different graphical elements in the long output from Aspose, mostly consisting of lines and arrowheads; the low-level encoding in the resulting PDF file, as shown in figures 2.5 and 2.6, were also nearly identical to the Image PDFs in Dr Wright's `TC` folder.

198. The only apparent difference was the coordinates on the page where the images were placed, and the scaling. Having considered these, I think it very likely that both files (Dr Wright's Images, and the Aspose automatic conversion) were indeed exactly the same up to a possible translation and scaling factor: the reference point may have been different, and possibly the scale too. It would have been relatively easy to write a program to check this, but time did not allow for that.

4.3 Pandoc

199. Pandoc is a well-known and versatile tool that allows one to convert between many different formats. I made the same experiment as with Writer2LaTeX and obtained similar results. The structure of the LaTeX code is rather simple and did in particular not add commands to alter the metadata.

4.4 Conclusion on automatic conversion tools

200. There is nothing inherently unsuitable or untoward about using tools for automatic conversion between different formats. It would very well have been possible to have used Writer2LaTeX for creating parts of a LaTeX source file in 2008, and Dr Wright mentions that at paras. 61 to 64 of his eighth witness statement. I need however to point out a few facts: none of these tools sets metadata explicitly in the LaTeX source, as is observed in Dr Wright's source files. A file created with OpenOffice then converted with e.g. Writer2LaTeX, then compiled by LaTeX, will identify itself as having been produced with whichever underlying TeX engine was used, *not* OpenOffice. It also would not

[1] `https://products.aspose.app/pdf/conversion/pdf-to-tex`

result in a precise letter-for-letter typesetting of the same file (including placement of words, and typesetting of formulae) as that used in OpenOffice, and I would not expect that to happen in any case; the documentation of Writer2LaTeX mentions clearly that they want to advantage of LaTeX's typesetting capabilities, in particular with regards to mathematics.

201. However, there is also an indication that the Aspose tool could have been used to generate the diagrams in TikZ, from an extant PDF document. This is very significant in my opinion as an explanation for how those diagrams were created, and it would indicate that the LaTeX files would have been created from a PDF, and not the other way around. I have however not been able to confirm that point conclusively in the time I have had, but the resemblance I observed at 4.2 is nonetheless very strong.

DECLARATION

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. I do not consider that any interest affects my suitability as an expert witness on any issues on which I have given evidence.
4. 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 affects this.
5. I have shown the sources of all information I have used.
6. I have exercised reasonable care and skill in order to be accurate and complete in preparing this report.
7. 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.
8. I have not, without forming an independent view, included or excluded anything which has been suggested to me by others including my instructing lawyers.
9. I will notify those instructing me immediately and confirm in writing if for any reason my existing report requires any correction or qualification or my opinion changes.
10. 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

and has done in this case;

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;

d. I may be required to attend Court to be cross-examined on my report; 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.

11. 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.
12. I confirm that I have acted in accordance with the Code of Practice for Experts.
13. 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: DocuSigned by: Arthur Rosendahl CE82BDF80AA841E...

Dated: 18/1/2024

List of Exhibits

{H/324}
- **Exhibit AR1** : Curriculum Vitae

{H/325}
- **Exhibit AR2** : Bitcoin White Paper - March 2009 version

{H/326}
- **Exhibit AR3** : Bitcoin White Paper - November 2008 version

{H/327}
- **Exhibit AR4** : Bitcoin White Paper - October 2008 version

{H/328}
- **Exhibit AR5** : Adhatarao, S. and Lauradoux, C., *"Robust PDF files forensics using coding style"*, IFIP International Conference on ICT Systems Security and Privacy Protection. Springer, Cham, 179-195(2022)

{H/329}
- **Exhibit AR6** : Plain text extracted from Candidate **A**

{H/330}
- **Exhibit AR7** : Plain text extracted from Candidate **B**

{H/331}
- **Exhibit AR8** : Plain text extracted from Candidate **C**

{H/332}
- **Exhibit AR9** : Plain text extracted from Candidate **D**

{H/333}
- **Exhibit AR10** : Plain text extracted from Candidate **E**

{H/334}
- **Exhibit AR11** : Plain text extracted from Candidate **F**

{H/335}
- **Exhibit AR12** : Plain text extracted from Candidate **G**

{H/336}
- **Exhibit AR13** : Plain text extracted from Candidate **H**

{H/337}
- **Exhibit AR14** : Plain text extracted from Candidate **I**

{H/338}
- **Exhibit AR15** : Plain text extracted from Candidate **J**

{H/339}
- **Exhibit AR16** : Plain text extracted from Candidate **K**

{H/340}
- **Exhibit AR17** : Plain text extracted from Candidate **L**

{H/341}
- **Exhibit AR18** : Plain text extracted from Candidate **M**

{H/342}
- **Exhibit AR19** : Plain text extracted from Candidate **N**

{H/343} • **Exhibit AR20** : Compiled version of Candidate **B**

{H/344} • **Exhibit AR21** : Compiled version of Candidate **D**

{H/345} • **Exhibit AR22** : Compiled version of Candidate **E**

{H/346} • **Exhibit AR23** : Compiled version of Candidate **F**

{H/347} • **Exhibit AR24** : Compiled version of Candidate **G**

{H/348} • **Exhibit AR25** : Compiled version of Candidate **H**

{H/349} • **Exhibit AR26** : Compiled version of Candidate **I**

{H/350} • **Exhibit AR27** : Compiled version of Candidate **J**

{H/351} • **Exhibit AR28** : Compiled version of Candidate **K**

{H/352} • **Exhibit AR29** : Compiled version of Candidate **M**

{H/353} • **Exhibit AR30** : Compiled version of Candidate **N**

{H/354} • **Exhibit AR31** : Print-out of `https://tug.org/pipermail/xetex/2008-November/011213.html`

{H/355} • **Exhibit AR32** : Print-out of `https://tug.org/TUGboat/tb30-2/tb95reutenauer.pdf`

{H/356} • **Exhibit AR33** : Print-out of `https://ctan.org/ctan-ann/id/mailman.2092.1276003629.2324.ctan-ann@dante.de`

{H/357} • **Exhibit AR34** : Print-out of `https://github.com/pgf-tikz/pgf/commit/a30f8b3f8dc285980c20e1638b9b25c4d00efe8d`

{H/358} • **Exhibit AR35** : Print-out of `https://ctan.org/ctan-ann/id/mailman.2919.1266100191.20360.ctan-ann@dante.de`

{H/359} • **Exhibit AR36** : Print-out of `https://ctan.org/ctan-ann/id/mailman.4502.1267396362.20360.ctan-ann@dante.de`

{H/360} • **Exhibit AR37** : Print-out of `https://askubuntu.com/questions/888225/installation-of-miktex`