International Handbook of Blockchain Law

A Guide to Navigating Legal and Regulatory Challenges of Blockchain Technology and Crypto Assets

Second Edition

EDITED BY
MATTHIAS ARTZT
THOMAS RICHTER

Wolters Kluwer

Summary of Contents

Table of Contents

Figure 1.4 An Example of an If ... Else Statement

```
if (age < 20 and age > 12)
{
    print "this person is a teenager"
}
else
{
    print "this person is not a teenager"
}
```

Another often used control structure is the *while loop* which allows the repeated execution of a section of logic for as long as a condition is *true*. Here is a *while loop* example which displays an integer variable as long as it is not 11, as it is being incremented by one, starting from 1 (Figure 1.5).

Figure 1.5 An Example of a While Statement

```
variable = 1
while (variable < 11)
{
    print variable
    variable = variable + 1
}
```

Control structures can be used to build ever more complex functionality from very basic building blocks,[25] which can then be stored as *functions* or routines and re-used in other computer programs. For example, it is very easy to build a multiplication function (seeking to multiply x and y) given an existing simple addition function by simply adding y to y, x number of times. Once such a multiplication function is created and can be accessed as a function (or as a *callable* set of code), it can then be used to create another yet more intricate function to calculate the *power* of a number, and so on.

All programming languages have developed extensive sets of such complex functions, procedures and reusable code called *libraries*. Libraries can be accessed, or called, by any computer program written in the applicable programming language without having to re-invent a large set of functions and procedures. This is one of the

25. Which build on the very basic capacities of the core circuitry of a computer's Central Processing Unit or CPU. A CPU has only a limited set of core functionalities, including addition and multiplication. The power of a CPU is in being able to do billions of these basic operations per second.

most important features of modern programming – the ability to re-use and build on existing code.

In many cases, sets of libraries for a particular program also include functionality known as an *Application Program Interface* or API, which allows other programming languages, and in some cases devices with programmatic functionality, to interact with programs written in the language for which the API is available.[26]

[3] Data Structures

Data structures are methods of storing data in computer memory in a particular way to enable certain capabilities for manipulating that data, which would be (much) more difficult or not possible without that data structure.

In many instances, data structures incorporate or enable certain relationships between data elements which then facilitate indexing, sorting, editing, or other functionality, including certain kinds of calculations. Data structures in many cases also help with the efficient access and manipulation of data stored in memory.

[a] The Array

An *array* is an example of a simple data structure, which is simply a list of items in order. Here is an example of an array containing the months of the year (Figure 1.6).

Figure 1.6 An Array of the Months of the Year

Jan.	Feb.	Mar.	Apr.	May	Jun.	Jul.	Aug.	Sep.	Oct.	Nov.	Dec.
0	1	2	3	4	5	6	7	8	9	10	11

The location of an element in an array is referred to as its *index*, displayed above below each month.[27]

An array is useful if one wants to relate a specific location in that array with the ranked order of the data items stored. In the above example, the months of the year are thus listed in the order in which they occur.

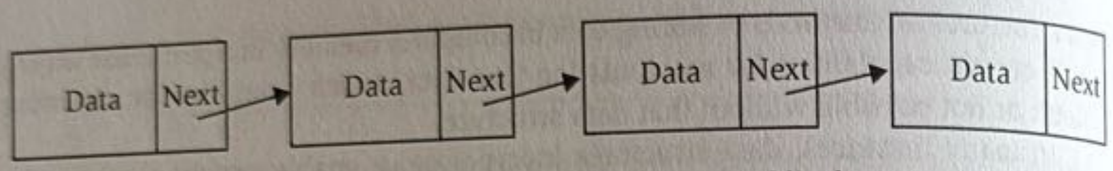
Another example is the list of ingredients in packaged food, usually found on its label. This array is usually required to be listed in the order of prevalence of the ingredients listed. The ingredient with the highest percentage of content goes first, the ingredient with the next highest goes next, and so on. The structure of the array thus embodies a sorted list without needing to do anything further.

26. *See*, for instance, the API for the Java programming language at: https://docs.oracle.com/en/java/javase/21/index.html (accessed 10 Jan. 2024).
27. Note that many programming languages start the index at 0, which is perhaps a bit counterintuitive but makes it easier for the computer, which deems 0 to be just as good a number as 1.

[b] The Linked List

The *linked list* is another example of a data structure. A linked list is a set of data objects where each data object, in addition to storing some data value, keeps track of the location of the next item in the list by way of a pointer (usually pointing to the memory address of the next item in the list) (Figure 1.7).

Figure 1.7 A Linked List



Linked lists are very useful for maintaining an ordered list because it is very easy to insert or remove elements from the list. For example, to insert a new element in an ordered list, one only needs to update the pointer of the previously ordered item in the list. Compare this to an array where one would need to shift all subsequent list members over one location to the right.

The core idea behind the linked list (tying data elements together with a pointer) is also used in a slightly different way in a blockchain.

[4] Syntax

The syntax of a programming language is the set of rules for translating a logical process or workflow into a sequence of instructions which the computer can understand and execute.

When writing a program in a specific language, it is very important to fully understand that language's syntax. Any misunderstanding of the syntax can lead to errors in the instructions which may go unnoticed in the final implementation of a program until a set of conditions arises which causes the execution of the program to go awry. These types of errors are often referred to as 'bugs', and no computer program is without them.[28]

It should be noted at this point that bugs or errors in how computer programs run can occur not only from syntax errors in the code but also from such things as:

- errors in the operating system software of the computer, for instance, how the operating manages its memory;
- errors in the hardware configuration of the computer, for instance, how data traffic is managed on the motherboard of the computer;

28. With the exception perhaps of the code written to power the lunar missions in the late 1960s and early 1970s.

- logical errors in the code which fail to contemplate the full range of possible values for input variables to programmatic control structures and may thus cause unintended program flow and/or memory overflows[29] as a result.

No computer program of any complexity is without some bugs.

[5] From Program Code to Machine Code

A computer language, such as Java, Python or Visual Basic, is actually a large computer program written to make programming of computers easier for humans. These higher-level languages use simple English words to represent their syntax, and many have complex integrated editing tools with a great deal of features.[30]

The product of these high-level programming languages is a relatively easy-to-read text file, compliant with the structural formatting and syntax requirements of the specific programming language used. This file is not directly usable by the computer for execution, however, and must first be converted to *machine code* interpretable by the Central Processing Unit (CPU)[31] of the computer on which the code is to be executed.

The type of machine code required depends on the architecture of the CPU in question. Machine code for an Intel processor[32] is different from that for an ARM processor,[33] for example. The required machine code is determined by the *instruction set* for the CPU, which can be thought of as the complete list of available instructions which can be issued to the CPU by a computer program.

The program which translates the program language specific text file into machine code is called a *compiler*.[34] The product of a compiler is a long string of zeroes and ones called a *bitstream*, which in the case of a computer program is usually referred to as a *binary*.

In some cases, an intermediary code interpreter called a *virtual machine*, *process virtual machine*, or *runtime environment* sits between the programming language and the CPU.

Such a virtual machine is used to create the ability to execute a particular program on any CPU for which a virtual machine exists.[35] The Java programming language uses the *Java Virtual Machine*, or JVM, exactly for this purpose, and it is one of the reasons why this language is so popular: it can run on many different types of hardware.

29. For instance, by placing a value in a memory location which is too large for the reserved space, which can then result in overwriting the next memory location.
30. Referred to as Integrated Development Environments, or IDEs.
31. The 'brains' of the computer, which contain hardwired circuitry which can execute basic computer instructions at a very high rate.
32. Found in most desktop computers.
33. Advanced RISC Machines, which is used in many mobile and embedded devices.
34. This is a dramatic oversimplification of what a compiler does, but it is sufficient for our limited purposes.
35. A virtual machine thus allows a programming language to become *multi-platform*.

In the case of the JVM, the compiler does not generate binary code but rather a *bytecode* file, which the JVM then converts to machine code usable by the specific hardware being used.

In many cases, the binary or bytecode is transmitted to a computer or virtual machine for execution using hex encoding. As we will see, this is also how smart contracts are encoded and transmitted to the blockchain.

[E] Computer Architecture

Given all the above functionalities, a computer is physically organized with the following key components and associated functionalities:

(1) The Central Processing Unit (CPU), which executes instructions stored in memory.
(2) Memory, primarily of two types: Random-Access Memory (RAM), which is volatile[36] memory used for temporary storage of data and program instructions that the CPU is currently using, and secondary memory, which is non-volatile memory used for long-term storage, such as a hard drive.
(3) Registers, which are small, high-speed storage locations within the CPU used for temporarily holding data during processing.
(4) The Instruction Set Architecture (ISA), which is the set of commands or instructions that a CPU understands and can execute. These are pre-programmed into the CPU.
(5) Cache Memory, which is high-speed memory located between the CPU and RAM and which is used to store frequently accessed data and instructions for faster retrieval.
(6) The Bus System, which is a set of communication pathways that transfer data between components. They include the address bus (identifying memory locations), data bus (transferring actual data), and control bus (managing the operation of components).
(7) The Motherboard, which is the main circuit board that connects and facilitates communication between the CPU, memory, storage, and peripheral devices.
(8) A System Clock, which synchronizes the operations of the CPU and other components, ensuring that computational work is properly coordinated and sequenced.

In addition to these, a computer will also have a set of input devices, such as a keyboard, mouse, touch screen, of scanner, which allow users to interact with the computer by providing data or commands, as well as output devices, such as a monitor, printer, and speakers.

36. 'Volatile' in this context means that the storage is not persistent if the storage medium loses power.

[1] *Optimizing Computer Architecture*

Over the last number of decades, there has been a great deal of innovation in making computer architectures perform at ever faster rates. Among these, the most effective improvements have resulted from technologies which enable simultaneous processing of multiple instructions by the computer. These include:

- Pipelining or Parallel Processing, which usually involves breaking down the execution of instructions into multiple stages, allowing the CPU to work on different instructions at the same time.
- Multi-Core Processors, which is the installation of multiple integrated processor cores on a single computer chip. Such chips are designed to optimize parallel processing.

These parallel processing features are integral to the capabilities of today's computing hardware.

§1.03 THE MATHEMATICS OF BLOCKCHAIN

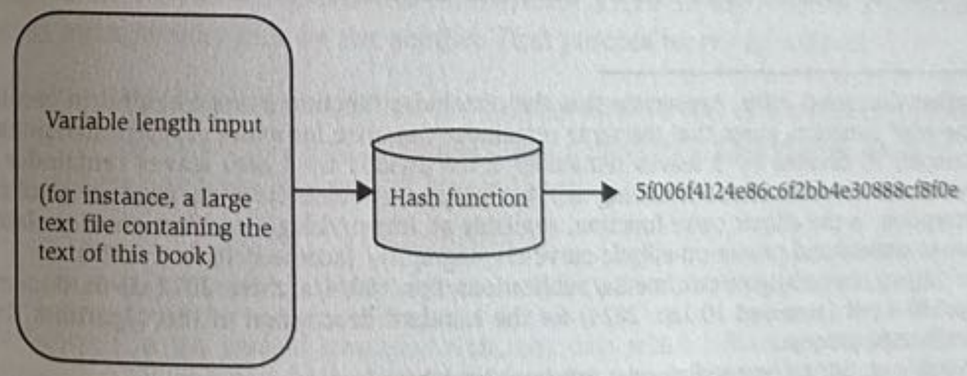
[A] The Hash Function

A function is a mathematical recipe for turning an input into a desired output based on a clearly articulated formula or algorithm. For example, the addition function 'C = A + B' takes the variables A and B as inputs and generates the desired output C by adding A and B together.

One group of functions relevant to understanding blockchain is the set of *hash functions*. Hash functions are used to convert a usually large input of data into a much smaller, fixed-length output to create a unique 'signature' of the larger input.

A hash function thus maps the input data to a numerical key. This key is usually referred to as the *hash value* or *hash digest*, or just the *hash* (Figure 1.8).

Figure 1.8 A Hash Function



Most hash functions work by taking the input data, chopping it up into smaller sections and then applying certain functions to those smaller chunks. The division and

remainder – or *modulo* – function is often used in this process, which gives a hash function a 'one-way' property: it is not possible to recreate the original input from the eventual hash generated by the function.[37]

These hash functions then shuffle and/or combine the outputs from these smaller chunks in a predefined manner and reapply the modulo, or some variant, again. This process is continued until a single output of the desired length is achieved.

The output of a hash function is often written in hex. For example, the very popular hashing algorithm SHA256[38] (which is used in the Bitcoin blockchain) will output a 64-bit hash digest which might look like this:[39]

5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8

Note that this hex value is the following very large number in base-10:

42,758,200,014,260,304,871,795,218,010,112,217,441,437,394,933,425,264,
440,882,844,866,153,524,052,696[40]

Hash functions are used in many applications, including storing passwords, digital certificates to support encryption, and indexing of large data sets. More refined hashing functions are used to assist in ensuring that data is not corrupted or has not been changed.

[B] The Cryptographic Hash Function

One specific type of hash function is particularly relevant to our understanding of blockchain: the *cryptographic hash function*. Like a regular hash function, the cryptographic hash function takes a data input of variable length and generates a fixed-length hash digest as its output.

A cryptographic hash function has some specific and very useful properties which the blockchain takes advantage of.

[1] The 'Always the Same' Property

A cryptographic hash function will always generate the same hash value for the same input. This is usually described by computer scientists and mathematicians as the

37. Further discussed *infra*. Appreciate that the remainder function is very helpful in creating a 'one-way' function, given that the same remainder can arise for many different divisions. For example, 10 divided by 3 leaves remainder 1. 16 divided by 5 also leaves remainder 1. A specialized implementation building on this property, which is often used in public key encryption, is the *elliptic curve* function, available at: https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/ (accessed 10 Jan. 2024).
38. *See* https://csrc.nist.gov/csrc/media/publications/fips/180/4/archive/2012-03-06/documents/fips180-4.pdf (accessed 10 Jan. 2024) for the standard description of this algorithm. It is a complicated process.
39. Available at: https://www.fileformat.info/tool/hash.htm (accessed 10 Jan. 2024).
40. This is a 78-digit number and one of the 2^{256} possible outputs from the SHA256 hashing algorithm. Note that the number of digits of that number can be determined by calculating $n \log 2$, in this case 256 log 2 which equals 78.

'deterministic' property. A function which is non-deterministic will generate different outputs because a certain amount of randomness exists in how the function works. An example of a non-deterministic function is the rolling of a set of dice.

[2] The 'Digital Signature' Property

A cryptographic hash function also uses a formula which ensures (with an extremely high level of confidence[41]) that the hash digest will be *unique*. A hash digest thus becomes a *digital signature* of the input data and can be used to determine whether an input has been modified or not.

In the use of a cryptographic hash function, the smallest possible change (i.e., one bit or character) in the input data will generate a completely different output digest (Figure 1.9).

Figure 1.9 An Example of How a Minimal Change in Input Generates a Completely Different Hash Digest

Input text	SHA256 hash digest
This is sample text submitted to the SHA256 hashing algorithm. We will change it just a little bit by adding a period to the end of this sentence below	bbadbf8c5138b8c9ced5c5197082ff5b36c9128f009bbb4cb450389c626c0214
This is sample text submitted to the SHA256 hashing algorithm. We will change it just a little bit by adding a period to the end of this sentence below	9d2f5d3b509a0334e752a0aaa0fe468f34affdddcc1c1912aeeee75321117514

This is a very valuable feature which enables a cryptographic hash function to serve several useful purposes. For example, this property is exploited when transmitting digital files over the internet to ensure they arrive on the receiver's computer in identical form to that sent by the sender. That process works as follows:

- The sender feeds the digital file (of any size) to one of a number of available cryptographic hashing functions[42] and generates a hash digest, usually somewhere between 32 and 128 bits in length.

41. Particularly in the case of *structured data*, i.e., data which follows a pattern, such as the grammar of a language. This property is sometimes also referred to as the function having 'strong collision resistance'.
42. For further review and testing with a number of these hashing algorithms: https://www.tools4noobs.com/online_tools/hash/ (accessed 10 Jan. 2024).

- The sender transmits the digital file to the recipient over the internet and also sends particulars of the cryptographic hash function used, together with the generated hash digest.
- The receiver feeds the received digital file to the cryptographic hash function, generating a hash digest.
- If the receiver's hash digest matches the hash digest sent by the sender, then we must conclude that the received file is identical to the one transmitted by the sender.
- If the hash digests do not match, then there was an error in transmission, and the file needs to be resent.

[3] The 'No Reverse Engineering' Property

A cryptographic hash function is also a 'one-way function' which means it is not possible to recreate or 'reverse engineer' the original file from the hash digest.

One might think that calculating all possible permutations of the input data and then looking for the digest in that list to find the original input data is an option. This is, however, computationally prohibitive – it would require computational power, and time, well beyond the capability of current computer technology – particularly with large input data sets of different sizes.[43]

One-way functions are very useful in many applications, including public-private key encryption.[44]

[C] The Hash List

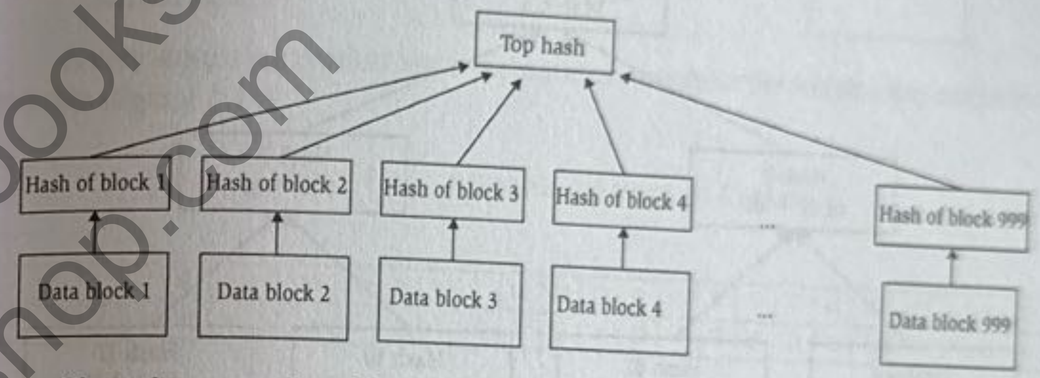
One application of a cryptographic hash function is the *hash list*. The hash list is used a lot in peer-to-peer (P2P) file sharing systems (*see* the P2P Networks section *infra*) to ensure that the various distributed pieces of a file come together to reconstitute the original file without errors. In that context, a hash list (Figure 1.10) functions as follows:

- The file being shared is broken up into several smaller pieces, called data blocks, which are sufficiently small to enable easy transmission over the internet as packets[45] or groups of packets.
- Each block is hashed, and the hash value is added to a separate hash list.
- Once all block hashes have been added, the hash list file is itself hashed and a *top hash* is generated.
- The top hash file is kept by a trusted authority.

43. These types of problems are called *intractable* which means they cannot be solved within the bounds of our foreseeable computational capacity, *see* https://en.wikipedia.org/wiki/Computational_complexity_theory#Intractability (accessed 10 Jan. 2024).
44. *See* the discussion *infra*.
45. A packet is a small block of data, easily transmissible between computers.

- Anyone wishing to obtain the file from the P2P network will first obtain the top hash file from the trusted source.[46]
- With that file in hand, the hash list can then be obtained from any other source (i.e., any other sharing user on the P2P network) for comparison with the top hash.
- If the hash of the hash list file matches the top hash, the user can proceed to download the various data blocks from any of the other users on the network who have those blocks available.
- Every block's hash is compared with the hash value for that block in the confirmed hash list file, and if it matches, it is accepted as genuine.

Figure 1.10 A Hash List



Given the properties of the cryptographic hash function, one is assured that if the hash of a received data block is not the same as the hash found in the hash list, a change or corruption is present in that block of data, necessitating a retransmission or obtaining the block from another source.

Using a hash list to verify the integrity of a file in blocks has another advantage in that it is much easier to determine where data errors exist than if one simply hashed the whole file. Fixing 'bad data' is thus much easier and faster.

[D] The Merkle Tree

A Merkle tree (also sometimes referred to as a *binary hash tree* or just *hash tree*) takes the ideas behind the hash list a step further for even better data integrity.

Rather than creating a simple list of hashes of data blocks, a Merkle tree incorporates data hashes into an upside-down tree-type structure, much like one might organize a family tree for a genealogy project. The individual elements in the tree are referred to as *nodes*, and every node has two children and one parent.

46. In the case of BitTorrent, this file actually contains both the top hash and the hash values for all the data blocks for the file being shared.