By now you will know that it is a good tradition of this blog to explain stuff by rebuilding toy examples of it in R (see e.g. [Understanding the Maths of Computed Tomography (CT) scans](http://blog.ephorie.de/understanding-the-maths-of-ct-scans), [So, what is AI really?](http://blog.ephorie.de/so-what-is-ai-really) or [Google’s Eigenvector… or how a Random Surfer finds the most relevant Webpages](http://blog.ephorie.de/googles-eigenvector-or-how-a-random-surfer-finds-the-most-relevant-webpages)). This time we will do the same for the hyped *Blockchain* technology, so read on!

Everybody is talking about blockchains, e.g. applications like the so-called *cryptocurrencies* (like *Bitcoins*) or *smart contracts* and the big business potential behind it. Alas, not many people know what the technological basis is. The truth is that blockchain technology, like any database technology, can be used for every conceivable content, not only new currencies. Business and governmental transactions as well as research results, data about organ transplants and items you gained in online games can be stored as can examination results and all kinds of certificates, the possibilities are endless. There are two big advantages:

* It is very hard to alter the content and
* you don’t need some centralized trustee.

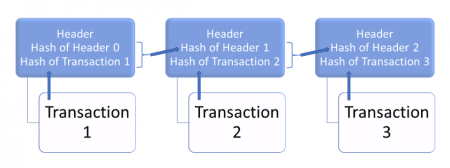
To understand why let us create a toy example of a blockchain in R. We will use three simple *transactions* as content:

trnsac1 <- "Peter buys car from Michael"

trnsac2 <- "John buys house from Linda"

trnsac3 <- "Jane buys car from Peter"

It is called a chain because the transactions are concatenated like so:



To understand this picture we need to know what a *hash* is. Basically, a hash (or better a *cryptographic hash* in this case) is just some function to encode messages. For educational purposes let us take the following (admittedly not very sophisticated) hash function:

# very simple (and not very good ;-)) hash function

hash <- function(x, l = 5) {

hash <- sapply(unlist(strsplit(x, "")), function(x) which(c(LETTERS, letters, 0:9, "-", " ") == x))

hash <- as.hexmode(hash[quantile(1:length(hash), (0:l)/l)])

paste(hash, collapse = "")

}

hash(trnsac1)

## [1] "104040200d26"

hash(trnsac2)

## [1] "0a1c2240401b"

We will use this function to hash the respective transaction and (and this is important here!) the *header* of the transaction before that. In this way, a header is created and the transactions form a chain (have a look at the pic again).

We will create the blockchain via a simple *data frame* but of course, it can also be distributed across several computers (this is why the technology is also sometimes called *distributed ledger*). Have a look at the function to add a transaction to an already existing blockchain or create a new one in case you start with NULL.

add\_trnsac <- function(bc, trnsac) {

if (is.null(bc)) bc <- data.frame(Header = hash(sample(LETTERS, 20, replace = TRUE)), Hash = hash(trnsac), Transaction = trnsac, stringsAsFactors = FALSE)

else bc <- rbind(bc, data.frame(Header = hash(paste0(c(bc[nrow(bc), "Header"]), bc[nrow(bc), "Hash"])), Hash = hash(trnsac), Transaction = trnsac))

bc

}

We are now ready to create our little blockchain and add the transactions:

# create blockchain

set.seed(1234)

bc <- add\_trnsac(NULL, trnsac1)

bc

## Header Hash Transaction

## 1 10050502060e 104040200d26 Peter buys car from Michael

# add transactions

bc <- add\_trnsac(bc, trnsac2)

bc <- add\_trnsac(bc, trnsac3)

bc

## Header Hash Transaction

## 1 10050502060e 104040200d26 Peter buys car from Michael

## 2 36353b35373b 0a1c2240401b John buys house from Linda

## 3 38383c1b391c 0a404040402c Jane buys car from Peter

To test the *integrity* of the blockchain we just recalculate the hash values and stop when they don’t match:

test\_bc <- function(bc) {

integrity <- TRUE

row <- 2

while (integrity && row <= nrow(bc)) {

if (hash(paste0(c(bc[(row-1), "Header"]), hash(bc[(row-1), "Transaction"]))) != bc[row, "Header"]) integrity <- FALSE

row <- row + 1

}

if (integrity) {

TRUE

} else {

warning(paste("blockchain is corrupted at row", (row-2)))

FALSE

}

}

# test integrity of blockchain

test\_bc(bc)

## [1] TRUE

Let us now manipulate a transaction in the blockchain! Mafia-Joe hacks his way into the blockchain and manipulates the second transaction so that not John but he owns Linda’s house. He even changes the hash value of the transaction so that it is consistent with the manipulated transaction:

# manipulate blockchain, even with consistent hash-value!

bc[2, "Transaction"] <- "Mafia-Joe buys house from Linda"

bc[2, "Hash"] <- hash("Mafia-Joe buys house from Linda")

bc

## Header Hash Transaction

## 1 10050502060e 104040200d26 Peter buys car from Michael

## 2 36353b35373b 0d0a332d271b Mafia-Joe buys house from Linda

## 3 38383c1b391c 0a404040402c Jane buys car from Peter

test\_bc(bc)

## Warning in test\_bc(bc): blockchain is corrupted at row 2

## [1] FALSE

Bingo, the integrity test cries foul! The consistency of the chain is corrupted and Mafia-Joe’s hack doesn’t work!

One last thing: in our toy implementation verifying a blockchain and creating a new one use the same amount of computing power. This is a gross oversimplification of what is going on in real-world systems: there creating a new blockchain is computationally much more expensive than verifying an existing one. For creating a new one huge amounts of possible hash values have to be tried out because they have to fulfill certain criteria (e.g. a number of leading zeros). This makes the blockchain extremely safe.

In the cryptocurrency world people (so-called *miners*) get paid (of course also in cryptocurrency) for finding valid hash values (called *mining*). Indeed big *mining farms* have been established which consume huge amounts of computing power (and therefore electricity, which is one of the disadvantages of this technology). For more details consult my question on Bitcoin.