WTF is The Blockchain?

The ultimate 3500-word guide in plain English to understand Blockchain.

Mohit Mamoria Follow
Jun 30, 2017 · 16 min read

http://www.forexnewsnow.com/top-stories/bitcoin-2016-summary-2017-forecasts/

Unless you're hiding under the rock, I am sure you'd have heard of Bitcoins and Blockchain. After all, they are the trending and media's favorite topics these days — the buzzwords of the year. Even the people who've never mined a cryptocurrency or understand how it works, are talking about it. I have more non-technical friends than technical ones. They have been bugging me for weeks to explain this new buzzword to them. I guess there are thousands out there who feel the same. And when that happens, there comes a time to write something to which everyone can point the other lost souls to — that's the purpose of this post — written in plain english that any regular internet user understands.

> *By the way, I am curator of a weekly newsletter, Unmade, which delivers one idea from the future to your inboxes.*

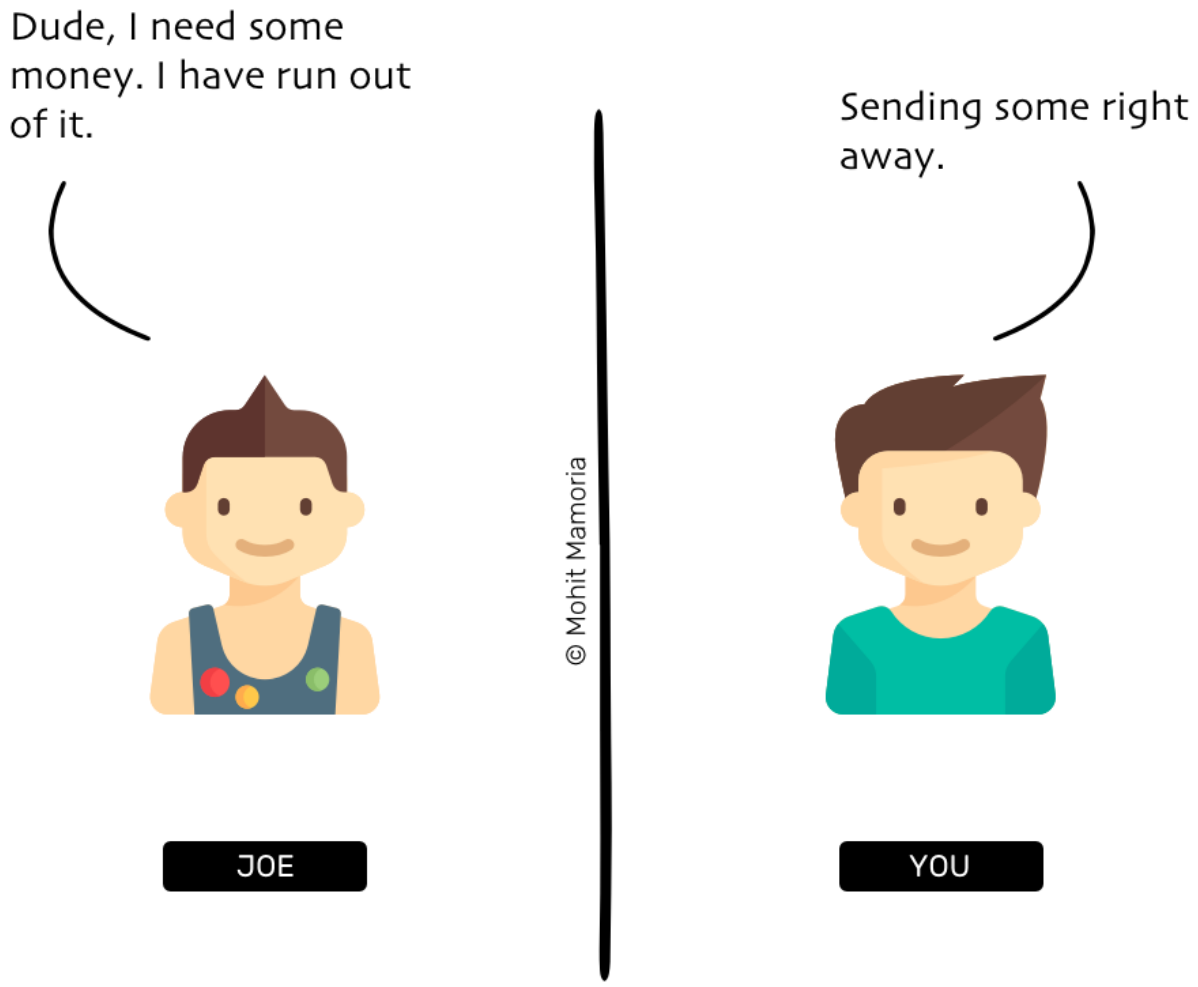
Blockchain: why do we even need something this complex?

> "For every complex problem there is an answer that is clear, simple, and wrong." — H. L. Mencken

Unlike every other post on the internet, instead of first defining the Blockchain, we'll understand the problem it solves.

Imagine, Joe is your best friend. He is traveling overseas, and on the fifth day of his vacation, he calls you and says, "Dude, I need some money. I have run out of it."

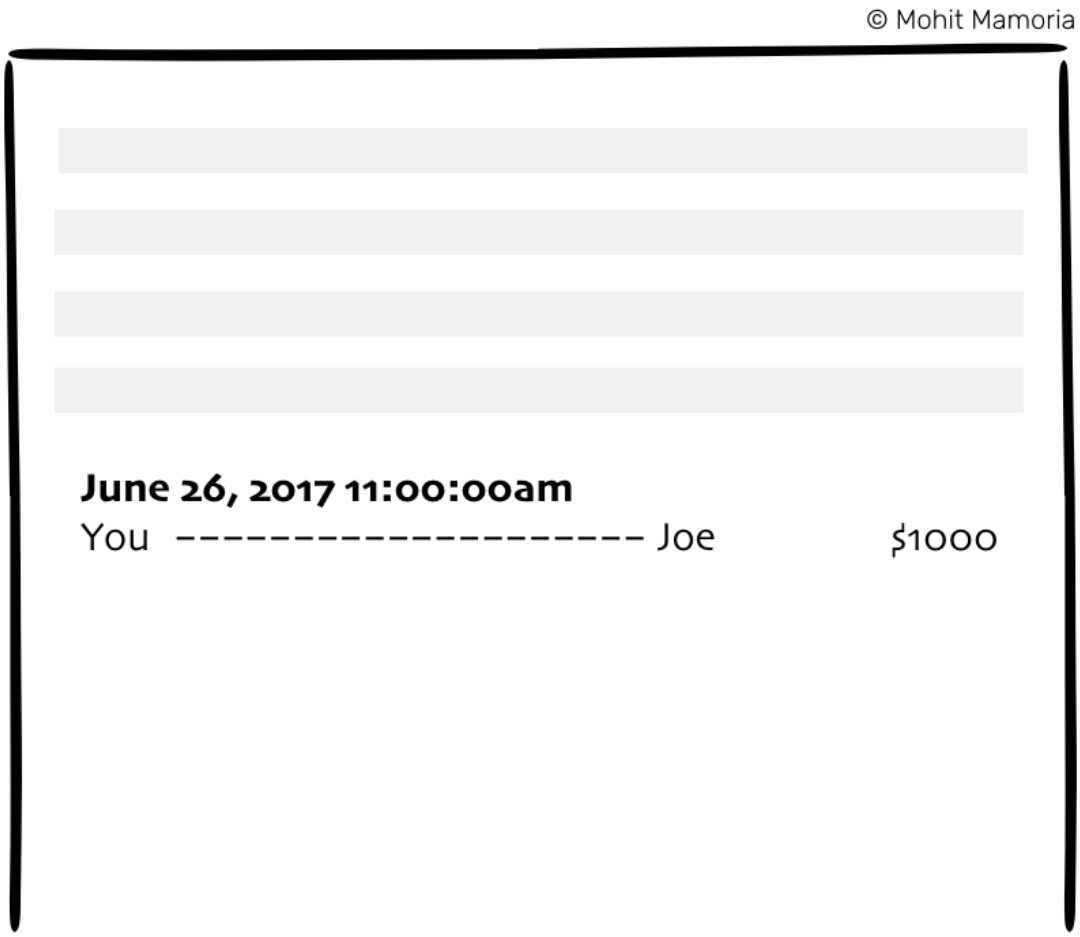
You reply, “Sending some right away,” and hung up.



You then call your account manager at your bank and tell him, “Please transfer $1000 from my account to Joe’s account.”

Your account manager replies, “Yes, sir.”

He opens up the register, checks your account balance to see if you have enough balance to transfer $1000 to Joe. Because you’re a rich man, you have plenty; thus, he makes an entry in the register like the following:



The Transaction Register

> ***Note:*** *We're not talking about computers only to keep things simple.*

You call Joe and tell him, "I've transferred the money. Next time, you'd go to your bank, you can withdraw the $1000 that I have just transferred."

What just happened? You and Joe both trusted the *bank* to manage your money. There was no real movement of physical bills to transfer the money. All that was needed was an entry in the register. Or more precisely, an entry in the register that neither you nor Joe controls or owns.

And that is the problem of the current systems.

> To establish trust between ourselves, we depend on individual third-parties.

For years, we've depended on these middlemen to trust each other. You might ask, "what is the problem depending on them?"

The problem is that they are singular in number. If a chaos has to be injected in the society, all it requires is one person/organization to go corrupt, intentionally or unintentionally.

- What if that register in which the transaction was logged gets burnt in a fire?
- What if, by mistake, your account manager had written $1500 instead of $1000?
- What if he did that on purpose?

> For years, we have been putting all our eggs in one basket and that too in someone else's.

Could there be a system where we can still transfer money without needing the bank?

To answer this question, we'll need to drill down further and ask ourselves a better question (after all, only better questions lead to better answers).

Think about it for a second, what does transferring money means? Just an entry in the register. The better question would then be —

> **Is there a way to maintain the register among ourselves instead of someone else doing it for us?**

Now, that is a question worth exploring. And the answer is what you might have already guessed. The blockchain is the answer to the profound question.

It is a method to maintain that register among ourselves instead of depending on someone else to do it for us.

Are you still with me? Good. Because now, when several questions have started popping in your mind, we will learn how this distributed register works.

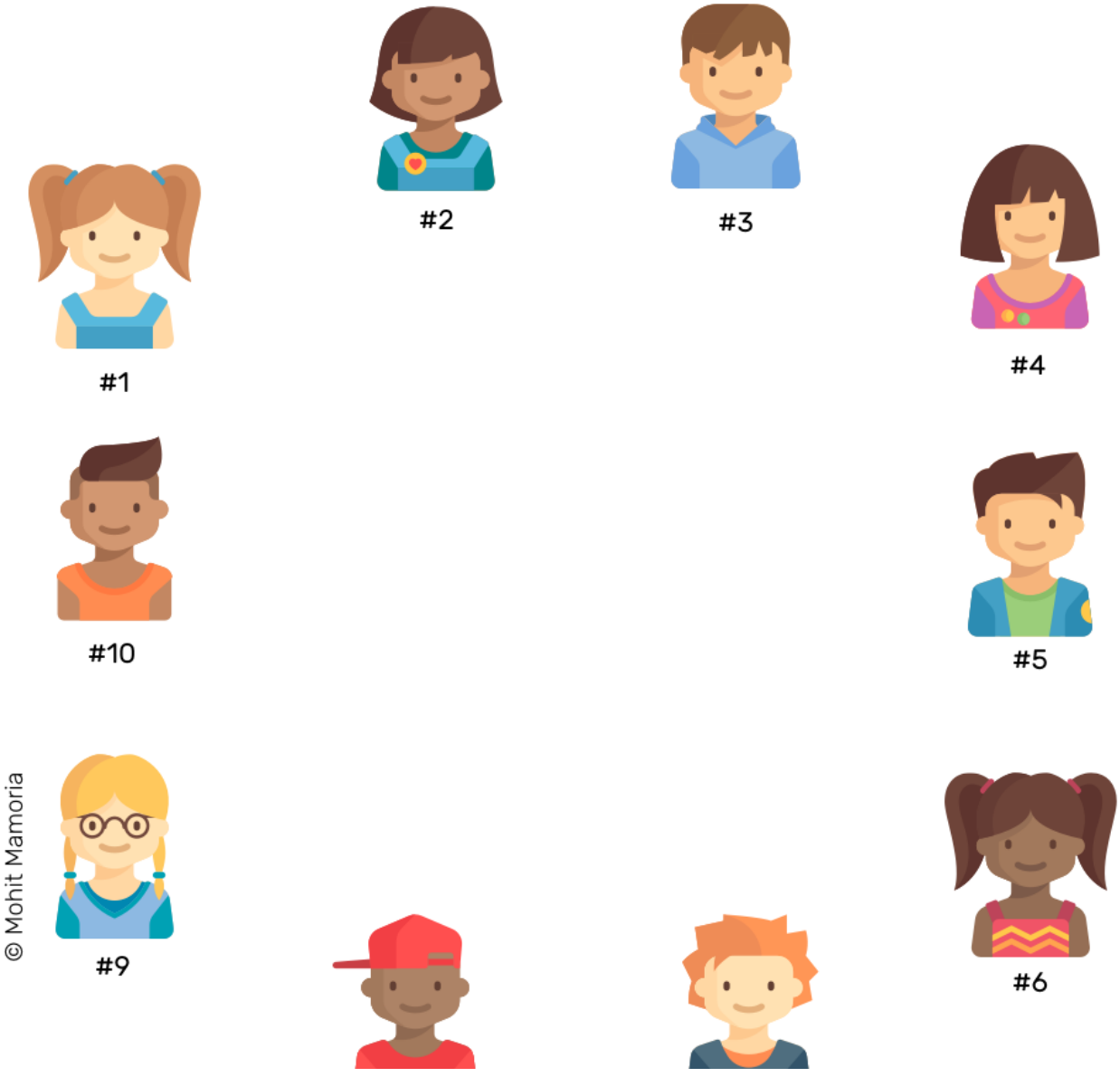
Yes, but tell me, how does it work?

The requirement of this method is that there must be enough people who would like not to depend on a third-party. Only then this group can maintain the register on their

own.

> “It might make sense just to get some Bitcoin in case it catches on. If enough people think the same way, that becomes a self-fulfilling prophecy.” — Satoshi Nakamoto in 2009

How many are enough? *At least three*. For our example, we will assume ten individuals want to give up on banks or any third-party. Upon mutual agreement, they have details of each other’s accounts all the time — without knowing the other’s identity.



1. An Empty Folder

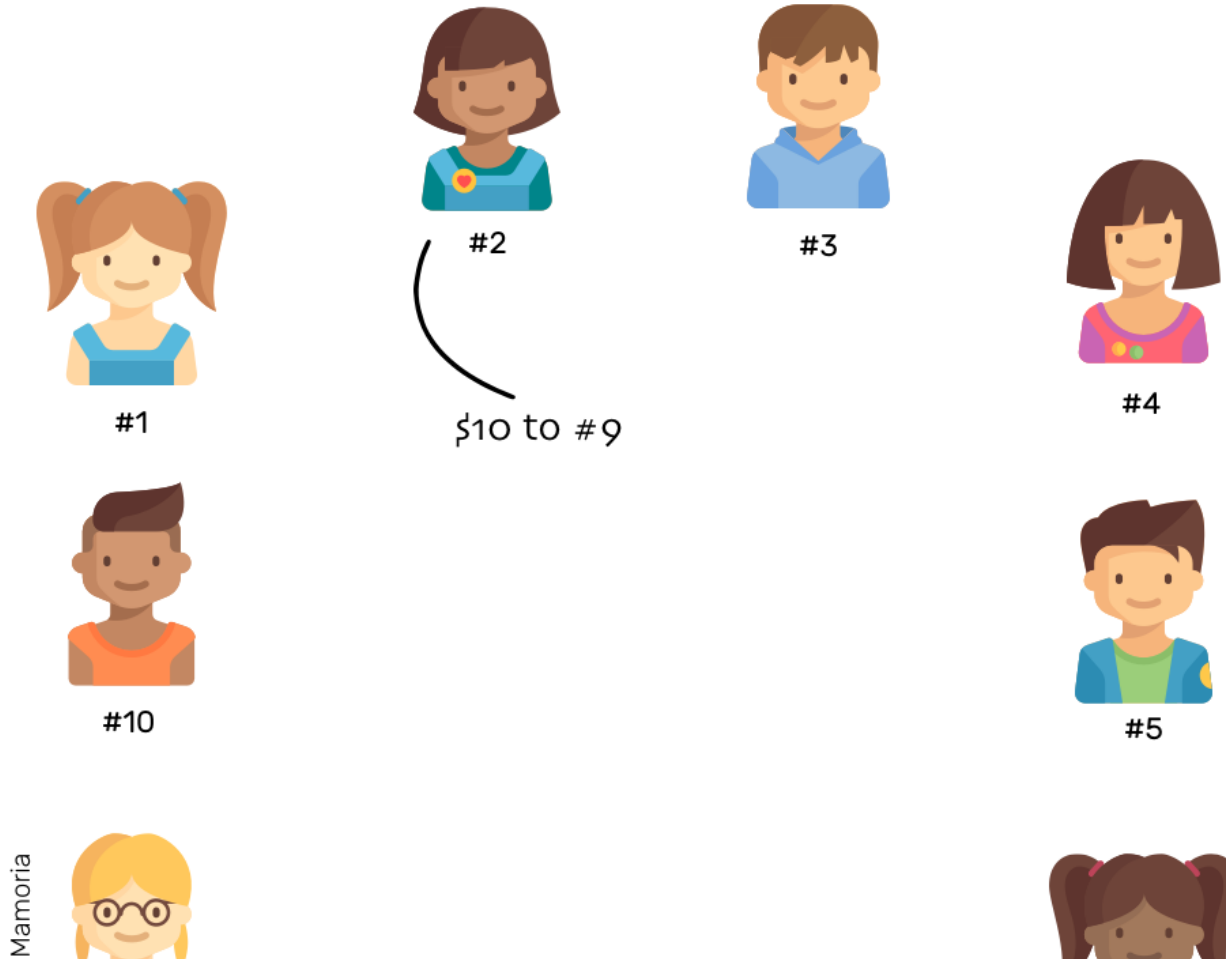
Everyone contains an empty folder with themselves to start with. As we'll progress, all these ten individuals will keep adding pages to their currently empty folders. And this collection of pages will form the register that tracks the transactions.

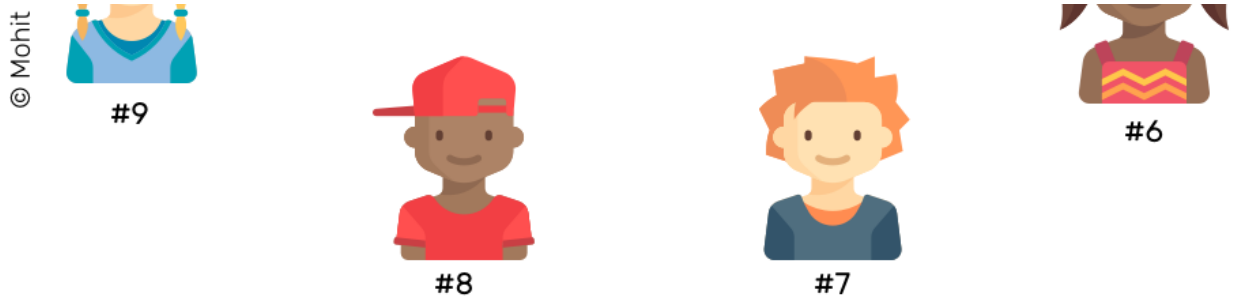
2. When A Transaction Happens

Next, everyone in the network sits with a blank page and a pen in their hands. Everyone is ready to write any transaction that occurs within the system.

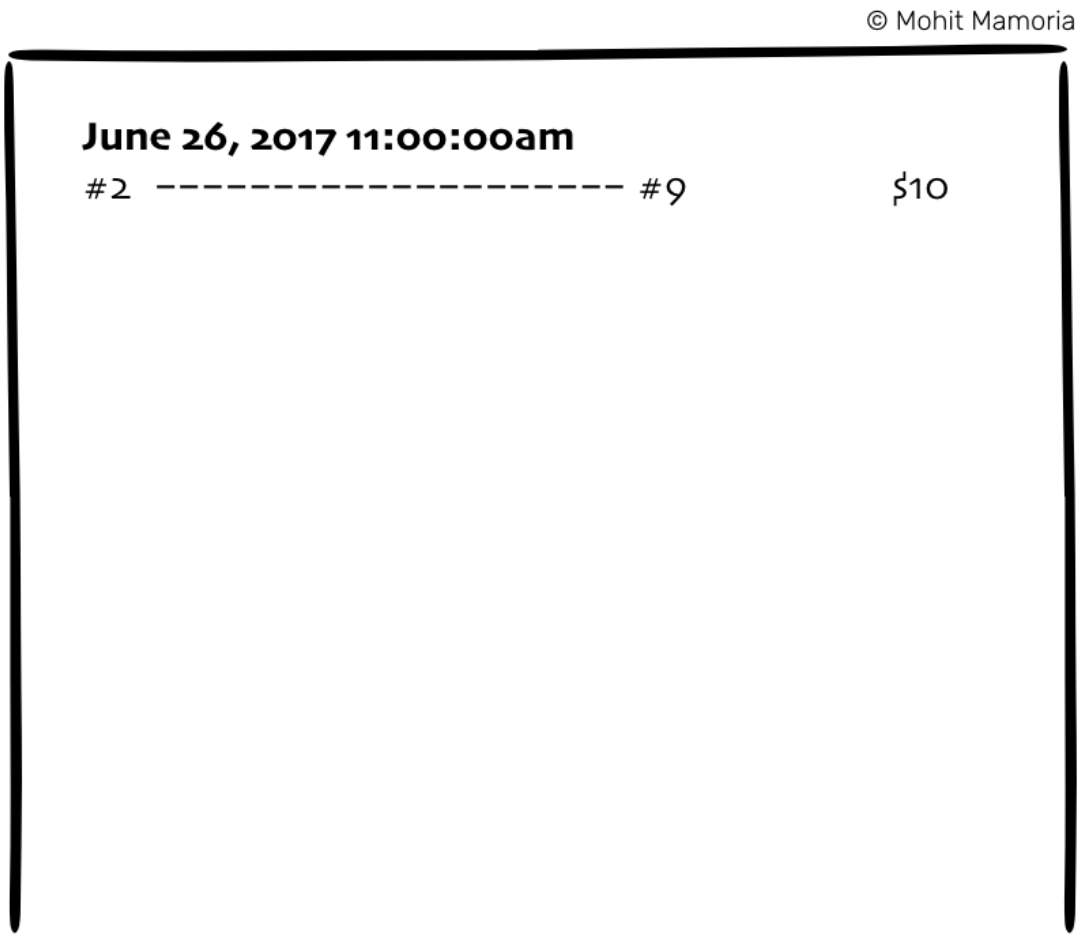
Now, if #2 wants to send $10 to #9.

To make the transaction, #2 shouts and tells everyone, "I want to transfer $10 to #9. So, everyone, please make a note of it on your pages."





Everyone checks whether #2 has enough balance to transfer $10 to #9. If she has enough balance, everyone then makes a note of the transaction on their blank pages.



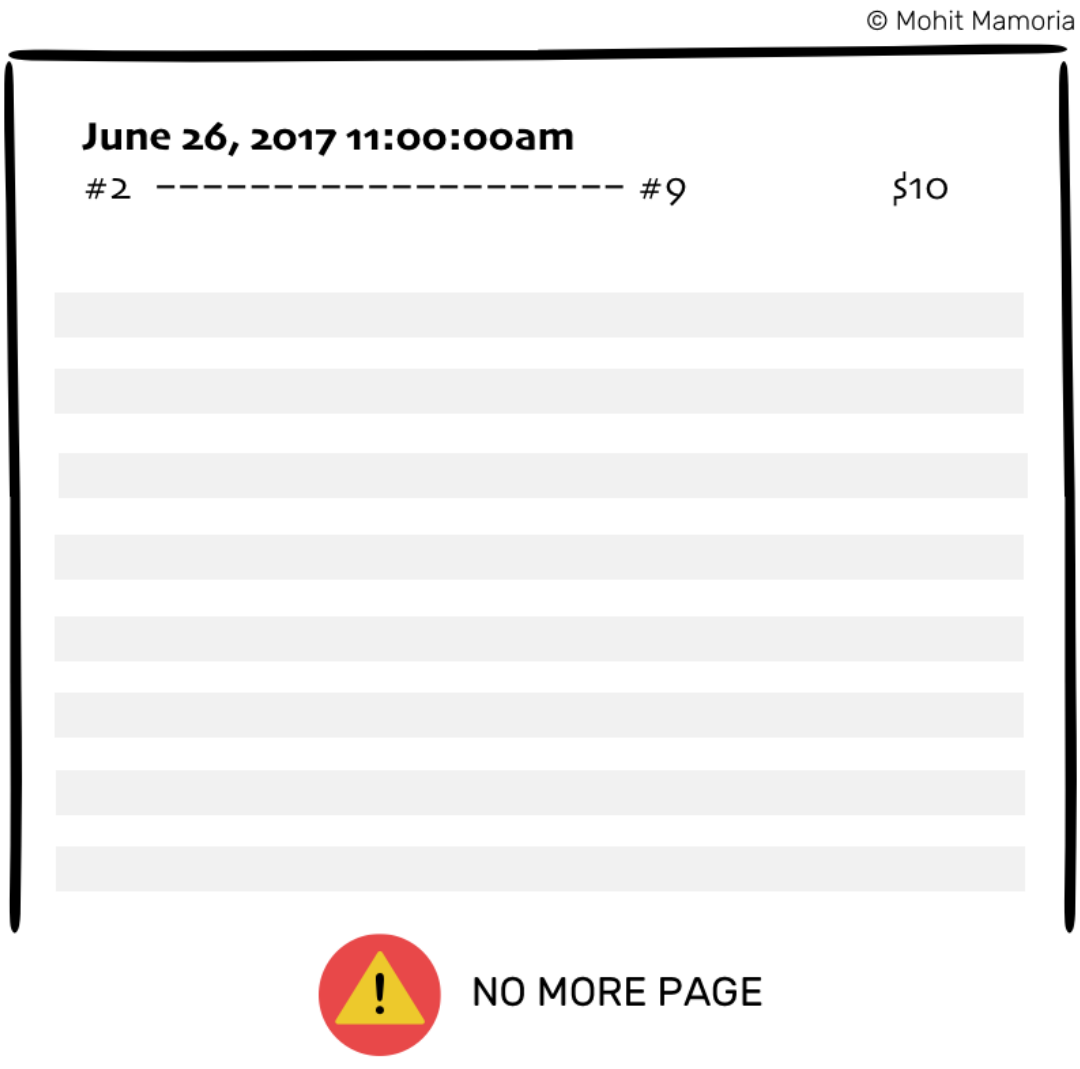
First transaction on the page

The transaction is then considered to be complete.

3. Transactions Continue Happening

As the time passes, more people in the network feel the need to transfer money to others. Whenever they want to make a transaction, they announce it to everyone else. As soon as a person listens to the announcement, (s)he writes it on his/her page.

This exercise continues until everyone runs out of space on the current page. Assuming a page has space to record ten transactions, as soon as the tenth transaction is made, everybody runs out of the space.



When page gets filled

It's time to put the page away in the folder and bring out a new page and repeat the process from the step 2 above.

4. Putting Away The Page

Before we put away the page in our folders, we need to *seal* it with a *unique key* that everyone in the network agrees upon. By sealing it, we will make sure that no one can make any changes to it once its copies have been put away in everyone's folder — not today, not tomorrow and not even after a year. Once in the folder, it will always stay in the folder — sealed. Moreover, if everyone trusts the seal, everyone trusts the contents of the page. And this sealing of the page is the *crux of this method.*

> **[Jargon Box]** *It is called 'mining' on the page to secure it, but for the simplicity of it, we'll keep calling it 'sealing.'*

> Earlier the third-party/middleman gave us the trust that whatever they have written in the register will never be altered. In a distributed and decentralized system like ours, this seal will provide the trust instead.

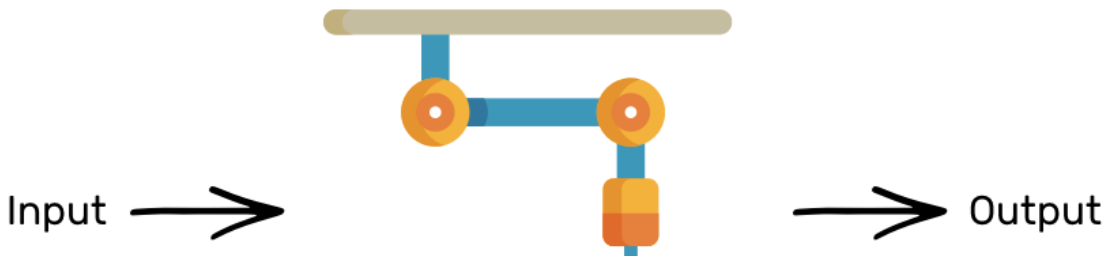
Interesting! How do we seal the page then?

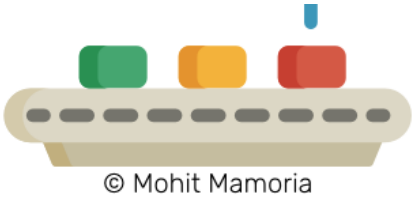
Before we learn how we can seal the page, we'll know how the seal works, in general. And as a pre-requisite to it is learning about something that I like to call…

The Magic Machine

Imagine a machine surrounded by thick walls. If you send a box with something inside it from the left, it will spit out a box containing something else.

> **[Jargon Box]** *This machine is called 'Hash Function,' but we aren't in a mood to be too technical. So, for today, these are 'The Magic Machines.'*

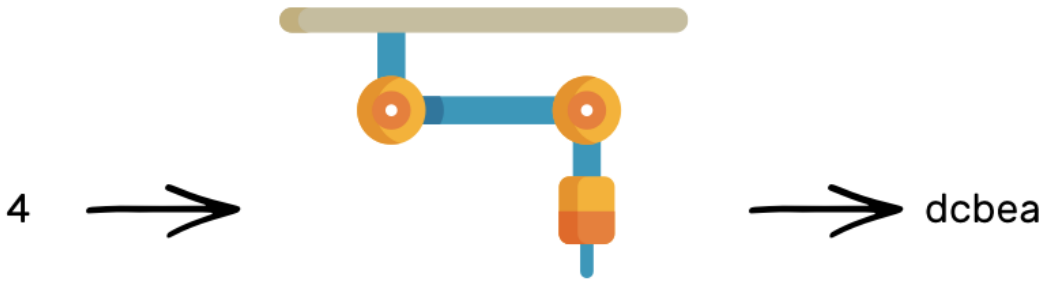




The Magic Machine (aka Hashing Function)

Suppose, you send the number 4 inside it from the left, we'd find that it spat out the following word on its right: 'dcbea.'

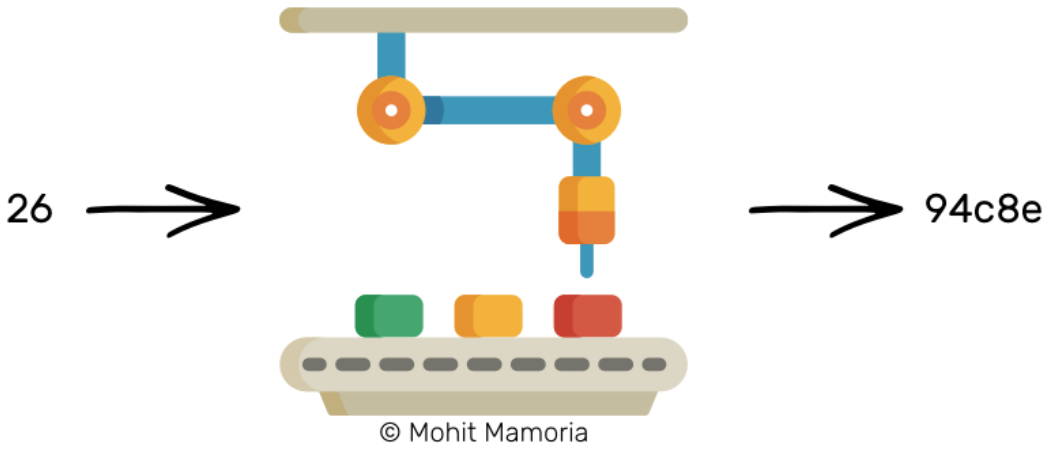
How did it convert the number 4 to this word? No one knows. Moreover, it is an irreversible process. Given the word, 'dcbea,' it is impossible to tell what the machine was fed on the left. But every time you'd feed the number 4 to the machine, it will always spit out the same word, 'dcbea.'



hash(4) == dcbea

Given the word, ‘dcbea,’ it is impossible to tell what the machine was fed on the left. But every time you’d feed the number 4 to the machine, it will always spit out the same word, ‘dcbea.’

Let’s try sending in a different number. How about 26?

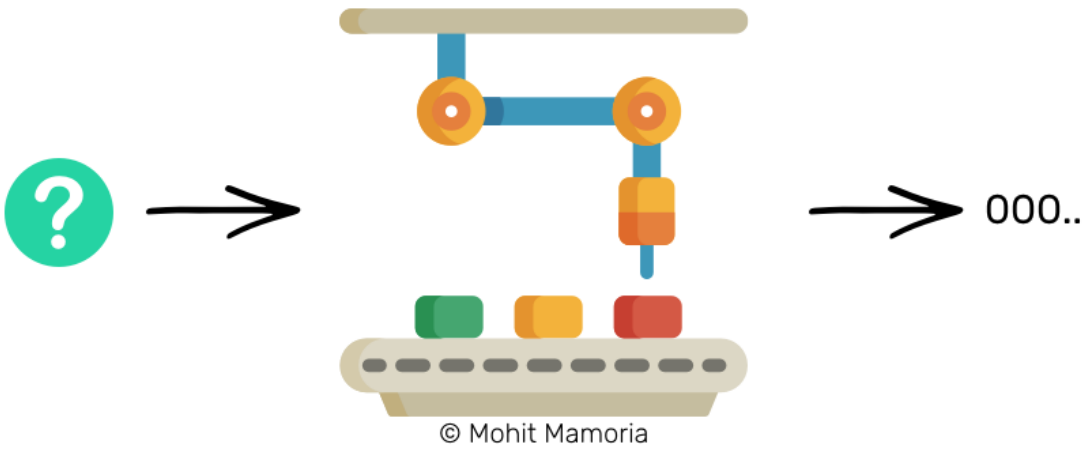


hash(26) == 94c8e

We got ‘94c8e’ this time. Interesting! So, the words can contain the numbers too.

What if I ask you the following question now:

> **“Can you tell me what should I send from the left side of the machine such that I get a word that starts with three leading zeroes from the right side of it? For example, 000ab or 00098 or 000fa or anything among the others.”**

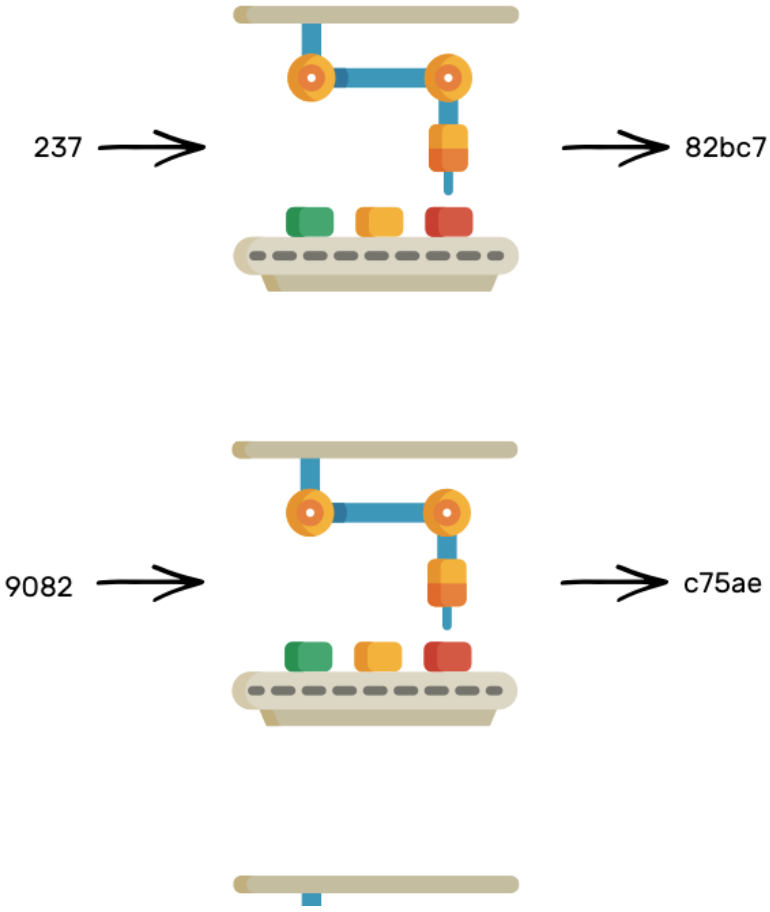


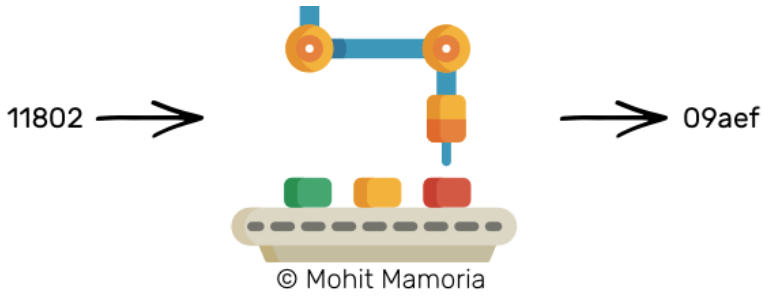
Predicting the input

Think about the question for a moment.

I've told you the machine has a property that we cannot calculate what we must send from the left after we're given the expected output on the right. With such a machine given to us, how can we answer the question I asked?

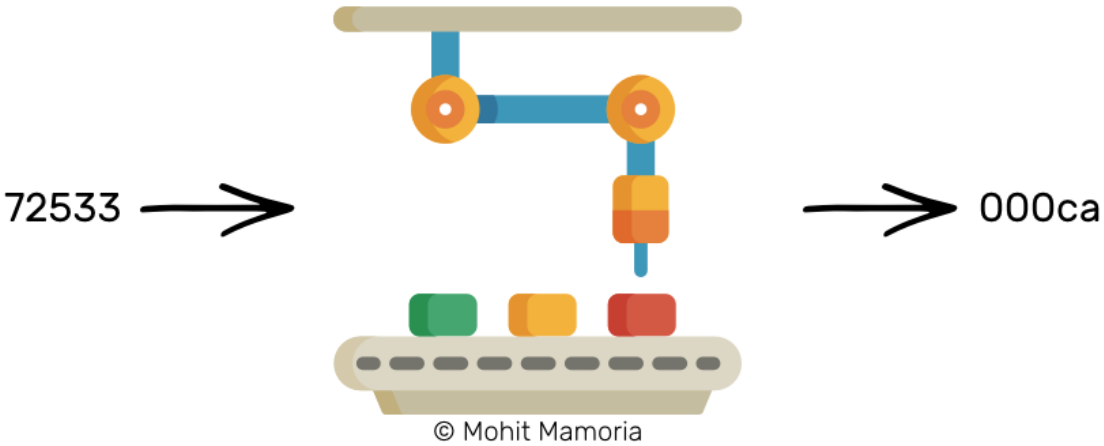
I can think of one method. Why not try every number in the universe one by one until we get a word that starts with three leading zeroes?





Try everything to calculate the input

Being optimistic, after several thousand attempts, we'll end up with a number that will yield the required output on the right.



It was extremely difficult to calculate the input given the output. But at the same time, it will always be incredibly easy to verify if the predicted input yields the required output. Remember that the machine spits out the same word for a number every time.

How difficult do you think the answer is if I give you a number, say 72533, and ask you the question, "Does this number, when fed into the machine, yields a word that starts with three leading zeroes?"

All you need to do is, throw the number in the machine and see what did you get on the right side of it. That's it.

The most important property of such machines is that — "Given an output, it is extremely difficult to calculate the input, but given the input and the output, it is pretty easy to verify if the input leads to the output."

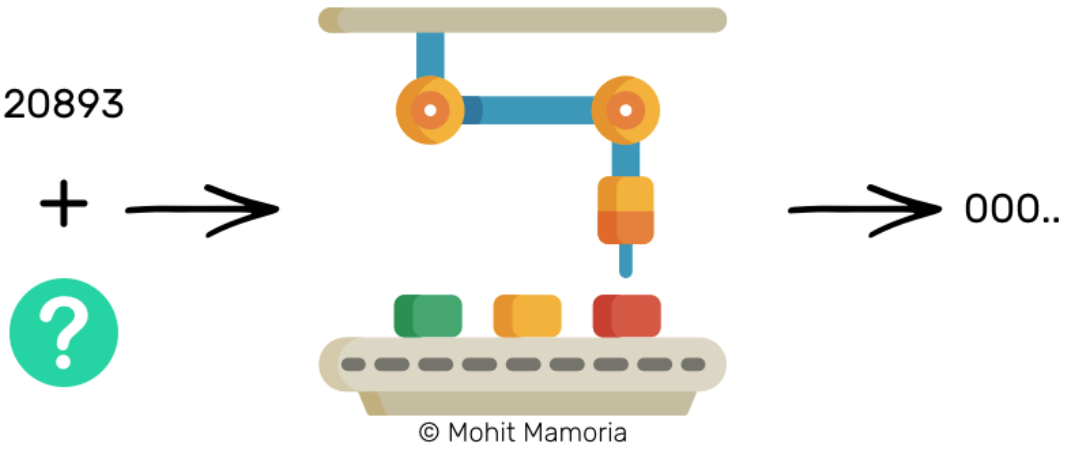
We'll remember this one property of the Magic Machines (or Hash Functions) through the rest of the post:

> Given an output, it is extremely difficult to calculate the input, but given an input and output, it is pretty easy to verify if the input leads to the output.

How to use these machines to seal a page?

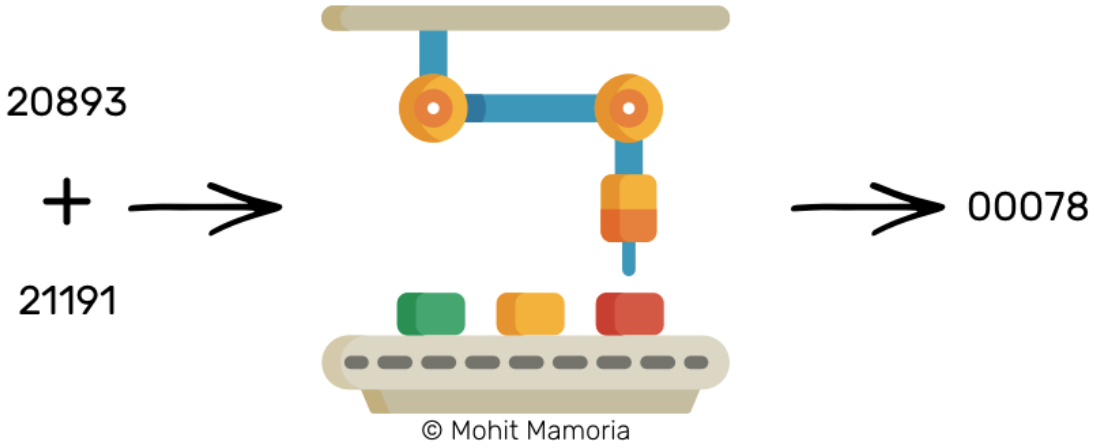
We'll use this magic machine to *generate a seal* for our page. Like always, we'll start with an imaginary situation.

Imagine I give you two boxes. The first box contains the number 20893. I, then, ask you, "Can you figure out a number that when added to the number in the first box and fed to the machine will give us a word that starts with three leading zeroes?"

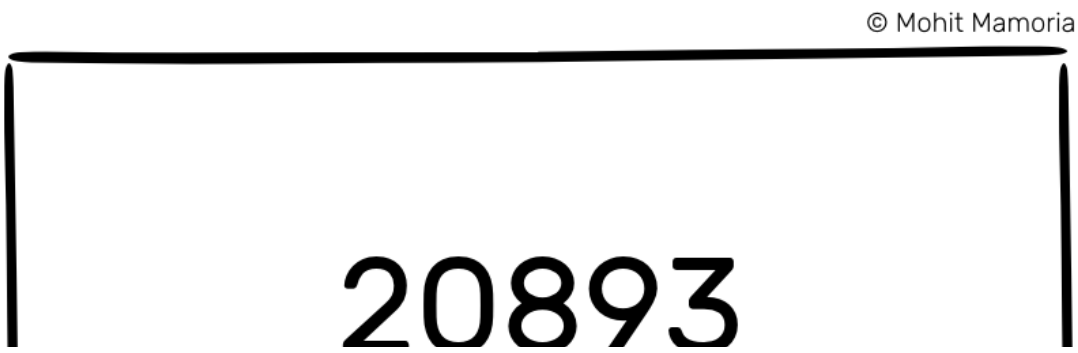


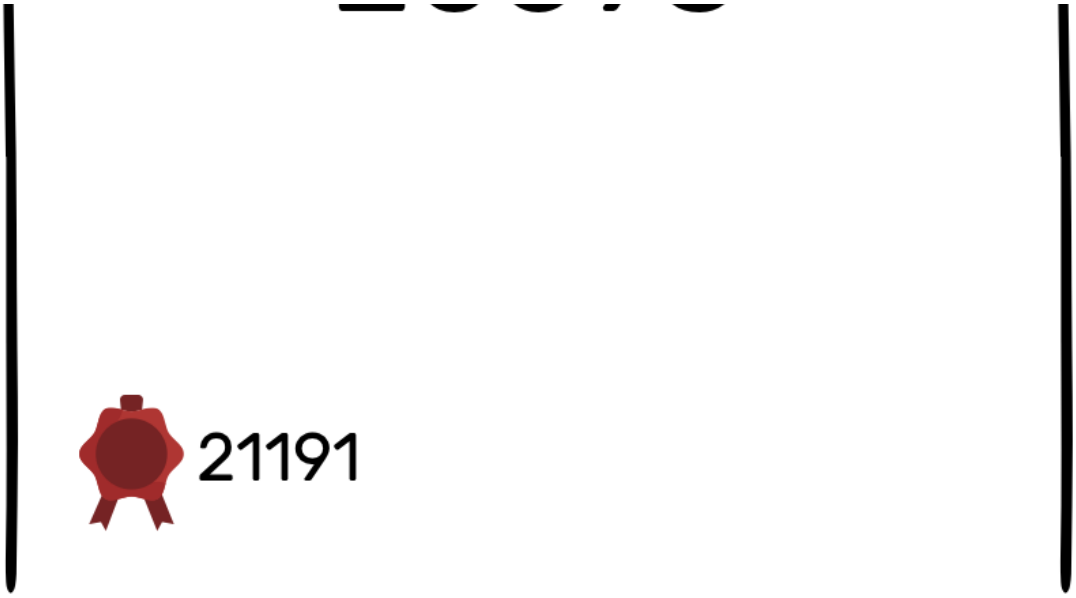
This is a similar situation as we saw previously and we have learned that the only way to calculate such a number is by trying every number available in the entire universe.

After several thousand attempts, we'll stumble upon a number, say 21191, which when added to 20893 (i.e. 21191 + 20893 = 42084) and fed to the machine, will yield a word that satisfies our requirements.



In such a case, this number, 21191 becomes the seal for the number 20893. Assume there is a page that bears the number 20893 written on it. To seal that page (i.e. no one can change the contents of it), we will put a badge labeled '21191' on top of it. As soon as the sealing number (i.e. 21191) is stuck on the page, the page is sealed.





The sealed number

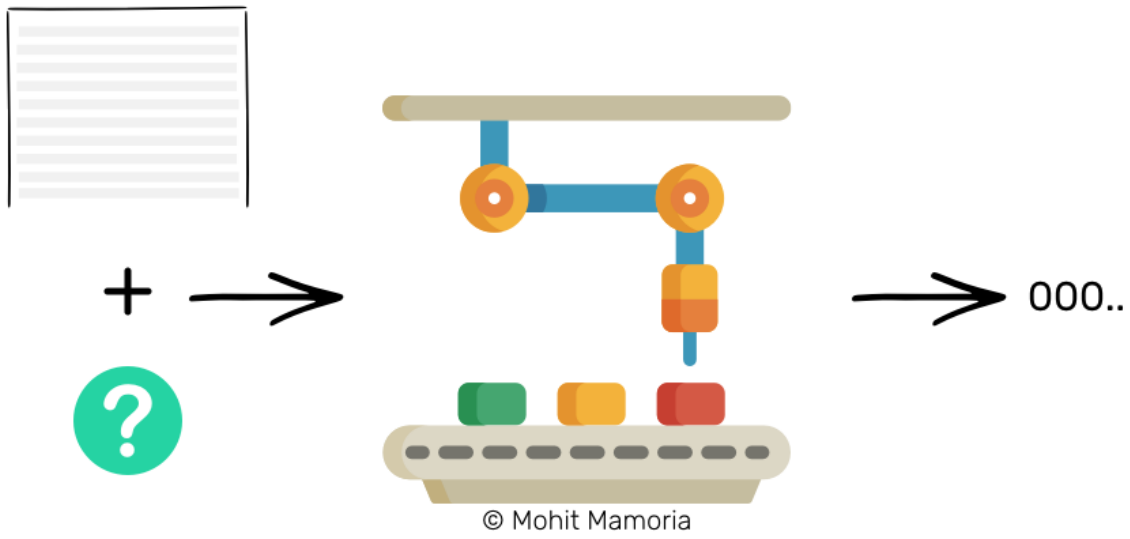
> **[Jargon Box]** *The sealing number is called 'Proof Of Work,' meaning that this number is the proof that efforts had been made to calculate it. We are good with calling it 'sealing number' for our purposes.*

If anyone wants to verify whether the page was altered, all he would have to do is — add the contents of the page with the sealing number and feed to the magic machine. If the machine gives out a word with three leading zeroes, the contents were untouched. If the word that comes out doesn't meet our requirements, we can throw away the page because its contents were compromised, and are of no use.

We'll use a similar sealing mechanism to seal all our pages and eventually arrange them in our respective folders.

Finally, sealing our page...

To seal our page that contains the transactions of the network, we'll need to figure out a number that when appended to the list of transactions and fed to the machine, we get a word that starts with three leading zeroes on the right.



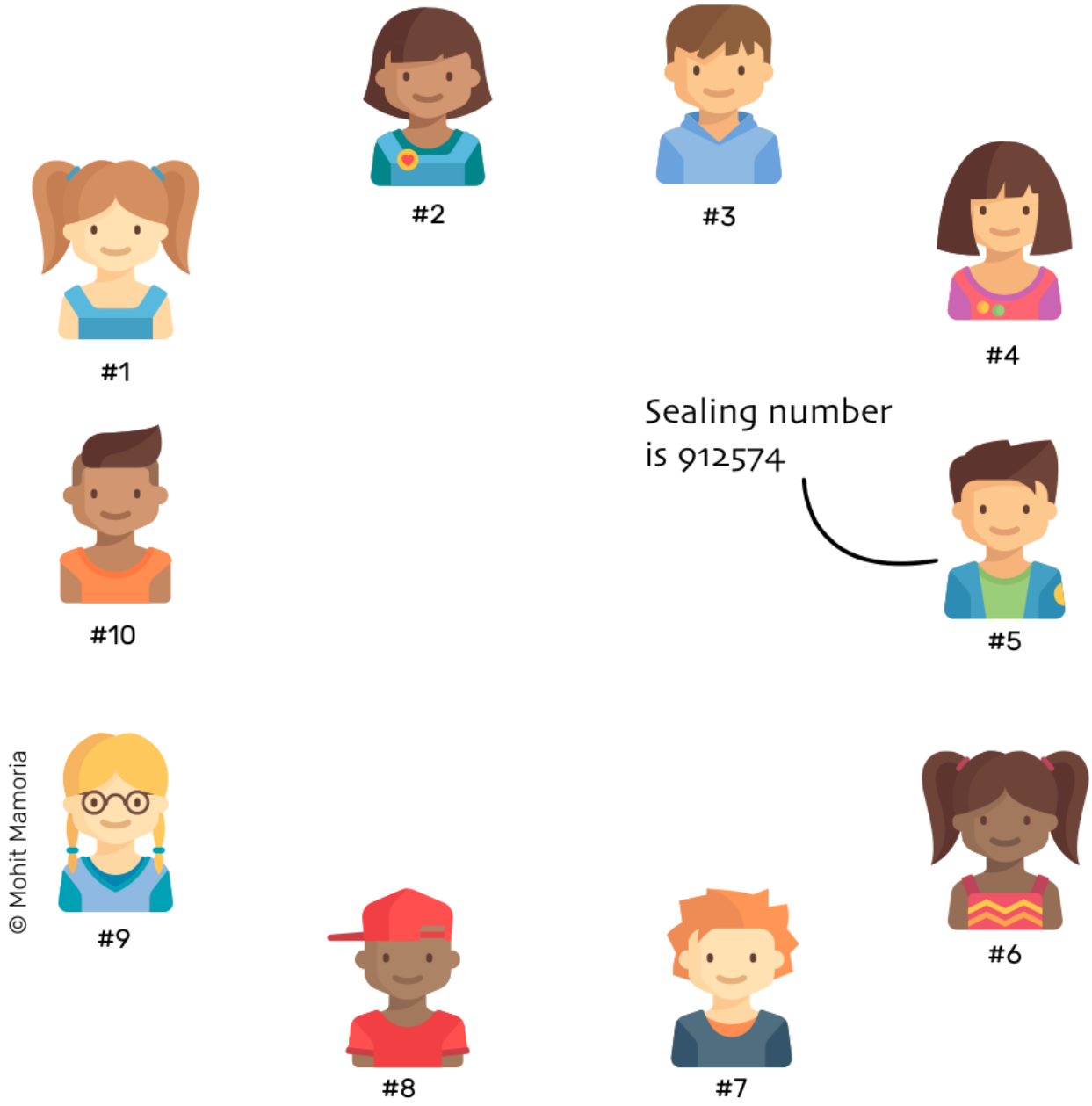
> ***Note:*** *I have been using the phrase 'word starting with three leading zeroes' only as an example. It illustrates how Hashing Functions work. The real challenges are much more complicated than this.*

Once that number is calculated after spending time and electricity on the machine, the page is sealed with that number. If ever, someone tries to change the contents of the page, the sealing number will allow anyone to verify the integrity of the page.

Now that we know about sealing the page, we will go back to the time when we had finished writing the tenth transaction on the page, and we ran out of space to write

more.

As soon as everyone runs out of the page to write further transactions, they indulge in calculating the sealing number for the page so that it can be tucked away in the folder. Everyone in the network does the calculation. The first one in the network to figure out the sealing number announces it to everyone else.



Immediately on hearing the sealing number, everyone verifies if it yields the required output or not. If it does, everyone labels their pages with this number and put it away in

their folders.

But what if for someone, say #7, the sealing number that was announced doesn't yield the required output? Such cases are not unusual. The possible reasons for this could be:

- He might have misheard the transactions that were announced in the network
- He might have miswritten the transactions that were announced in the network
- He might have tried to cheat or be dishonest when writing transactions, either to favor himself or someone else in the network

No matter what the reason is, #7 has only one choice — to discard his page and copy it from someone else so that he too can put it in the folder. Unless he doesn't put his page in the folder, he cannot continue writing further transactions, thus, forbidding him to be part of the network.

> Whatever sealing number the majority agrees upon, becomes the honest sealing number.

Then why does everyone spend resources doing the calculation when they know that someone else will calculate and announce it to them? Why not sit idle and wait for the announcement?

Great question. This is where the incentives come in the picture. Everyone who is the part of the Blockchain is eligible for rewards. The first one to calculate the sealing number gets rewarded with free money for his efforts (i.e. expended CPU power and electricity).

Simply imagine, if #5 calculates the sealing number of a page, he gets rewarded with some free money, say $1, that gets minted out of thin air. In other words, the account balance of #5 gets incremented with $1 without decreasing anyone else's account balance.

That's how Bitcoin got into existence. It was the first currency to be transacted on a Blockchain (i.e. distributed registers). And in return, to keep the efforts going on in the

network, people were awarded Bitcoins.

When enough people possess Bitcoins, they grow in value, making other people wanting Bitcoins; making Bitcoins grow in value even further; making even more people wanting Bitcoins; making them grow in value even further; and so on.

> The rewards make everyone keep working in the network.

And once everyone tucks away the page in their folders, they bring out a new blank page and repeat the whole process all over again — doing it forever.

> **[Jargon Box]** *Think of a single page as a Block of transactions and the folder as the Chain of pages (Blocks), therefore, turning it into a Blockchain.*

And that, my friends, is how Blockchain works.

. . .

Except that there's one tiny thing I didn't tell you. Yet.

Imagine there are five pages in the folder already — all sealed with a sealing number. What if I go back to the second page and modify a transaction to favor myself? The sealing number will let anyone detect the inconsistency in the transactions, right? What if I go ahead and calculate a new sealing number too for the modified transactions and label the page with that instead?

To prevent this problem of someone going back and modifying a page (Block) as well as the sealing number, there's a little twist to how a sealing number is calculated.

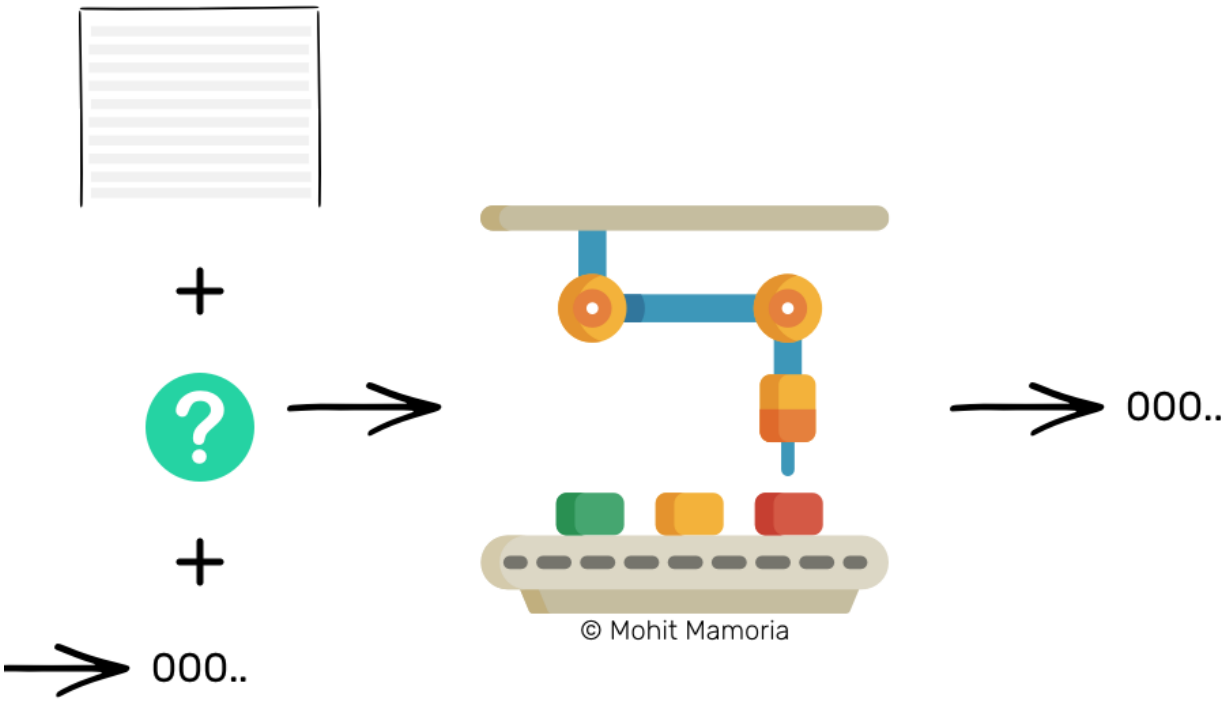
Protecting modifications to the sealing numbers

Remember how I told you that I had given you two boxes — one containing the number 20893 and another empty for you to calculate? In reality, to calculate the sealing

number in a Blockchain, instead of two boxes, there are three — two pre-filled and one to be calculated.

And when the contents of all those three boxes are added and fed to the machine, the answer that comes out from the right side must satisfy the required conditions.

We already know that one box contains the list of transactions and one box will contain the sealing number. The third box contains the output of the magic machine for the previous page.

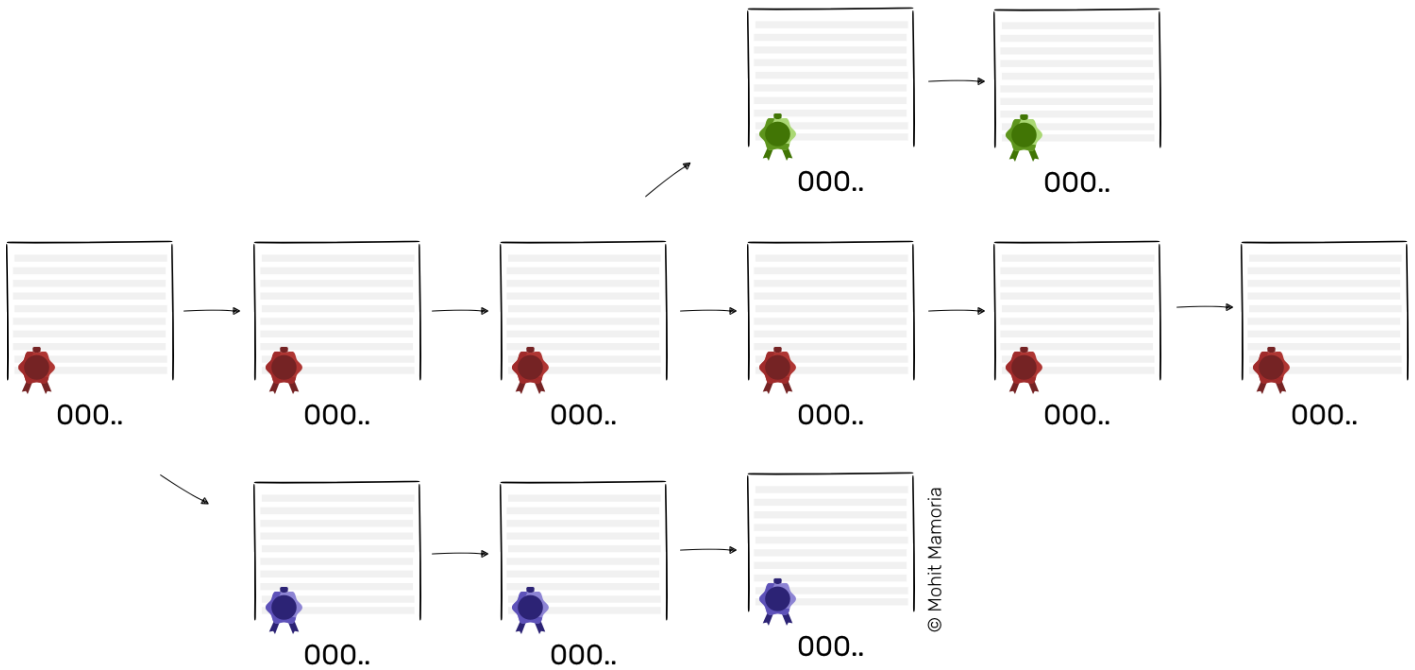


With this neat little trick, we have made sure that every page depends on its previous page. Therefore, if someone has to modify a historical page, he would also have to change the contents and the sealing number of all the pages after that, to keep the chain consistent.

If one individual, out of the ten we imagined in the beginning, tries to cheat and modify the contents of the Blockchain (the folder containing the pages with the list of transactions), he would have to adjust several pages and also calculate the new sealing numbers for all those pages. We know how difficult it is to calculate the sealing numbers. Therefore, one dishonest guy in the network cannot beat the nine honest guys.

What will happen is, from the page the dishonest guy tries to cheat, he would be creating another chain in the network, but that chain would never be able to catch up with the honest chain — simply because one guy's efforts and speed cannot beat cumulative efforts and speed of nine. Hence, guaranteeing that the longest chain in a network is the honest chain.

Longest chain is the honest chain.



Longest chain is the honest chain.

When I told you that one dishonest guy cannot beat nine honest guys, did it ring any bell in your head?

What if, instead of one, six guys turn dishonest?

In that case, the protocol will fall flat on its face. And it is known as "51% Attack". If the majority of the individuals in the network decides to turn dishonest and cheat the rest of the network, the protocol will fail its purpose.

And that's the only vulnerable reason why Blockchains might collapse if they ever will. Know that, it is unlikely to happen but we must all know the vulnerable points of the system. It is built on the assumption that the *majority of a crowd is always honest*.

And that, my friends, is all there is about Blockchains. If you ever find someone feeling left behind and wondering, "WTF is the Blockchain?" you know where you can point them to. Bookmark the link.

Can think of someone right now who should read this? The 'Share' button is all yours.

—

About the author

Mohit Mamoria is the curator of a weekly newsletter, Unmade, which delivers one idea from the future to your inboxes.

. . .

medium.com

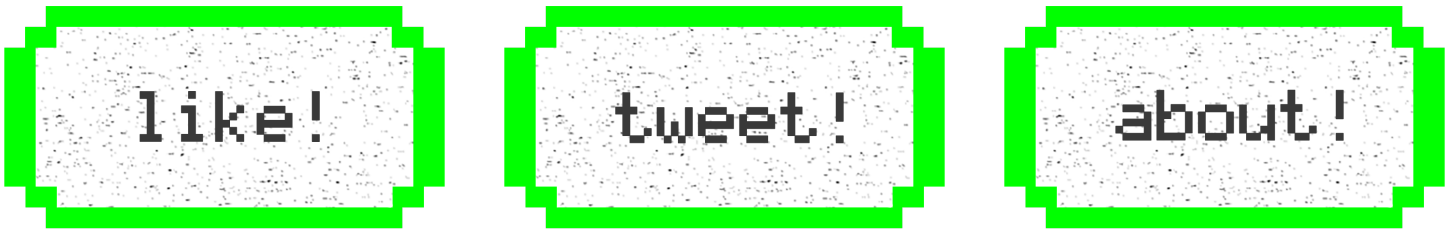
Blockchain and The Great Game of Attention

Everything you wanted to know about Blockchain, Tokens and their major use-cases

keepingstock.net

. . .

Thanks for reading! :) If you liked it, please support by hitting that heart button below. ♡



> *Hacker Noon is how hackers start their afternoons. We're a part of the @AMI family. We are now accepting submissions and happy to discuss advertising & sponsorship opportunities.*
>
> *If you enjoyed this story, we recommend reading our latest tech stories and trending tech stories. Until next time, don't take the realities of the world for granted!*

Bitcoin Blockchain Ethereum Inspiration Cryptocurrency