chainfrog

WHAT ARE SMART CONTRACTS?

WHAT ARE SMART CONTRACTS

AND WHERE AND WHY WOULD YOU USE THEM

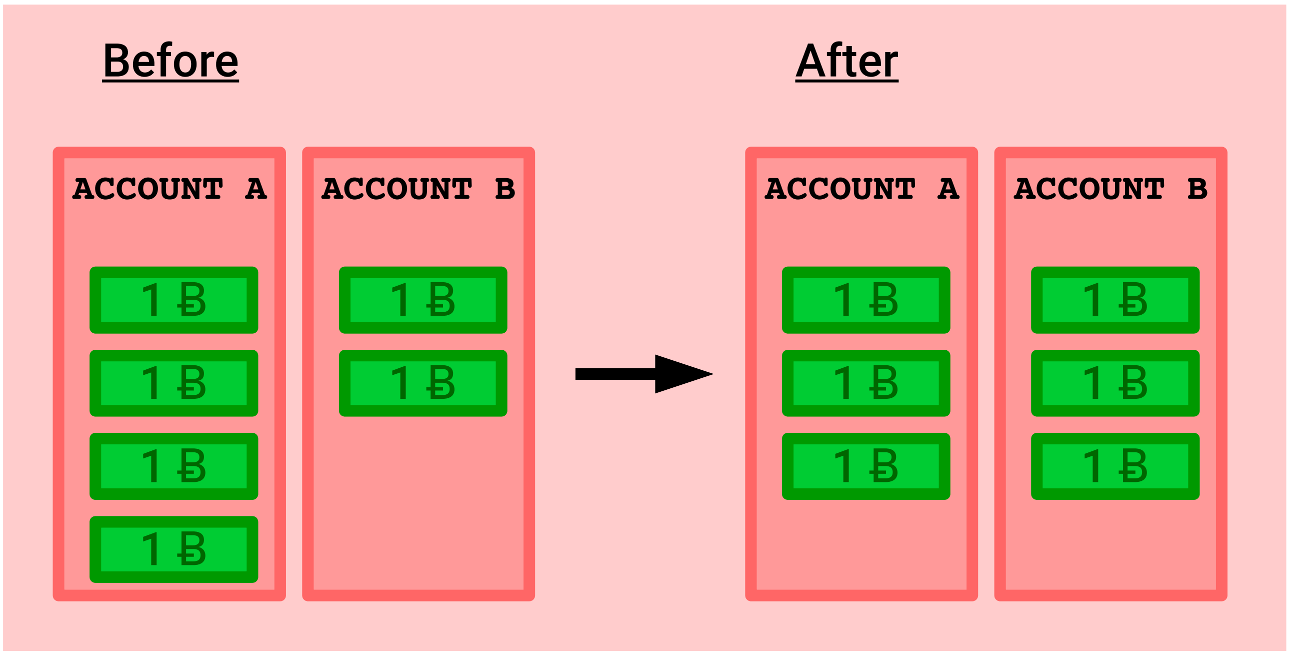
A question I get asked again and again at lectures and conferences is, “what exactly are smart contracts?”

“Are they going to replace conventional contracts? Are lawyers going to be ousted by programmers in the future? Can smart contracts fully automate the running of a company?”

It’s not the simplest concept to get your head around, but in this article I will describe what they are, what they can and can’t do, and what some of the risks associated with them are. We’ll start with the history of smart contracts and blockchains, get the technical part out of the way, and then look at the social side of smart contracts.

SMART CONTRACTS ON THE BLOCKCHAIN

If you have a public, open accounts ledger that is tamper-proof, with a system that allows anyone to create their own unique "bank account number", the simplest method for storing the balance of an account would be to treat the bank account number as a computer variable – each account number has a value, or balance. Transferring "coins" from one account to another would then simply involve subtracting the amount to transfer from one account, and adding the same amount to the second account. As long as everything balances out, things are good.

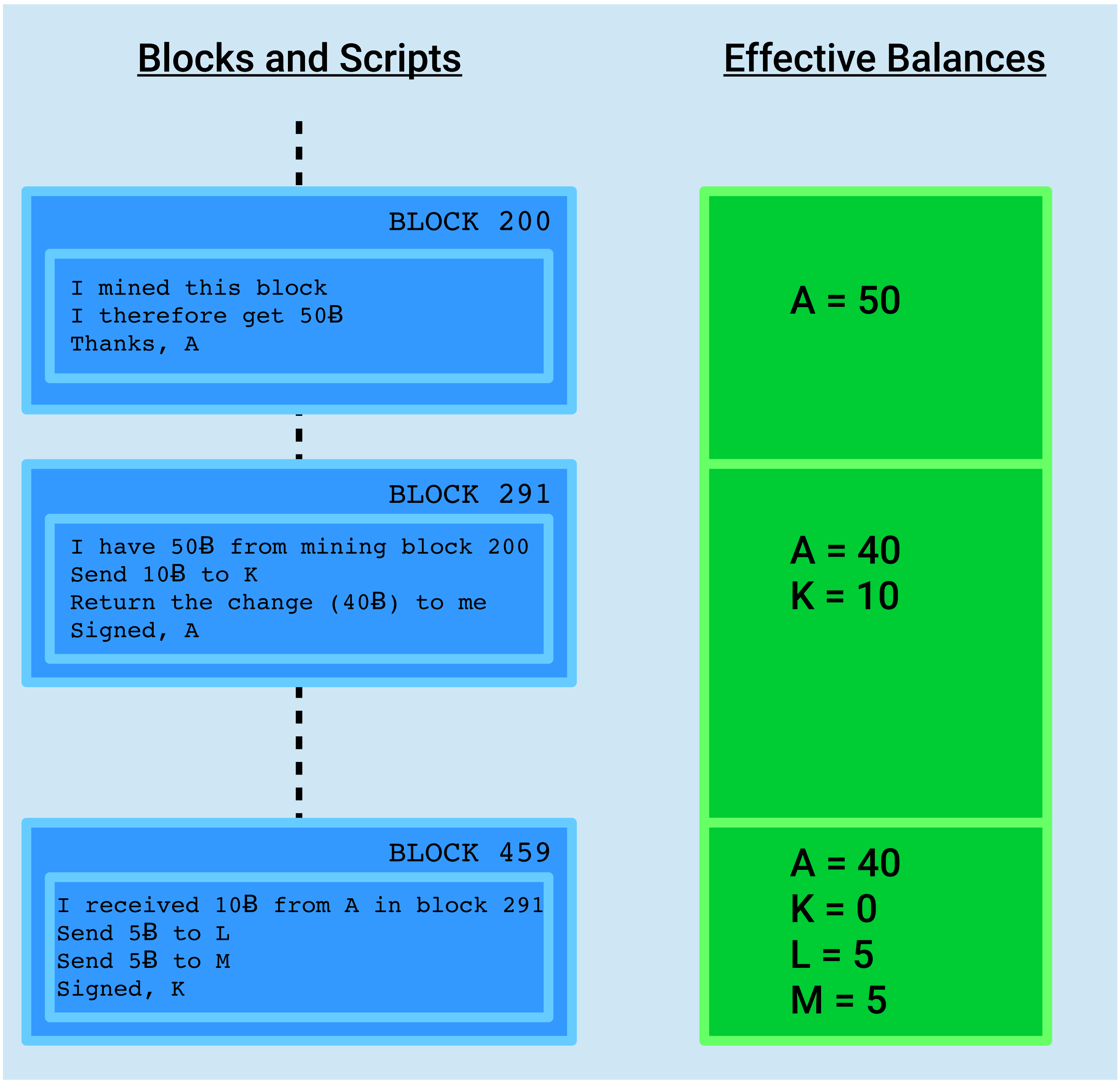


Bitcoin scripts

Satoshi Nakamoto – the inventor of Bitcoin – took a longer term view. He (or she, or they, as Nakamoto's true identity is unknown) considered the fact that with a public ledger, there might be a need for more complicated transactions than just “transfer X coins from A to B”. For example, what if you wanted to have transfers that require multiple parties to sign off on the validity of the transaction, or if you want to have transfers that can be redeemed after a set waiting period, or perhaps even some other esoteric transaction that no-one has thought of yet?

In 2009 when launching Bitcoin, Nakamoto implemented the value transfer system using a programming language that allows you to submit short programs to the peer-to-peer network, and which are run on each node within the network. In simpler terms: within the computer program running a Bitcoin node, there is a “virtual machine”: something similar to a tiny scientific calculator with a limited form of memory and some functions for handling the checking of digital signatures. When you transfer bitcoins from your address to someone else’s address, what you are actually doing is submitting a short simple computer program, known as a script, to the Bitcoin network and hence to copies of this tiny calculator on every node. The script is checked by each one to make sure it is a valid program and that it returns a value of “true” when run, and if it does the script is added to the blockchain.

At a later point, when the person you sent the bitcoins to wants to send them on to another address, they submit another script that references your earlier script, and as long as running their new script returns a results of “true”, the bitcoins are transferred onwards. The diagram below shows this in a simplified form (there are a heap of technicalities involved in the actual system, but we don’t need to discuss them here).



In a sense there really aren't actually any "bitcoins" being transferred from from one address to another. Instead, there is a chain of scripts scattered through the history of the blockchain, and each one when run returns the value "true". Effectively, there are thousands upon thousands of tiny computer programs stored on the blockchain, and when someone submits a new financial transaction, the relevant ones are retrieved and run again to check that the transaction is allowed.

These scripts are the first practical "smart contracts" on a blockchain. When you think about, a standing payment from your bank account to cover your electricity bill, or a subscription to a magazine, or a direct debit, are all smart contracts too!

Ethereum and Solidity

Fast forward to 2013, and a young Canadian/Russian programmer called Vitalik Buterin proposes a blockchain with a Turing-complete virtual machine in each node. Although Bitcoin has a couple of dozen commands in its scripting engine, it doesn't support loops or functions, and the scripts it can run are basically concerned with moving bitcoins about under certain condition. In Buterin's system, called Ethereum, the virtual machine that runs within each node doesn't just support simple scripts, it allows complete computer programs with loops, functions, user definable variables, and everything else a programmer needs. There's even a compiler, called Solidity, that allows coders to write in a Javascript-like language and compile their programs into the machine code that Ethereum runs.

But the problem with a Turing-complete machine is that there is no guarantee the programs submitted to it won't run forever. And a program that won't stop would tie up all the nodes on the peer-to-peer network if they tried to execute it. It's called the halting problem, and Ethereum has an ingenious solution to it. For your program to run on the blockchain, you need to supply "gas", which is paid for with Ether, the blockchain's associated cryptocurrency. Ether costs real money to buy, so programmers will only fuel up their programs with enough to achieve what they want, and if there's a bug, the program will burn through all the gas and just … stop.

Apart from the added complexity, Ethereum programs, or smart contracts, are just like Bitcoin scripts. They sit on the blockchain, and when the right conditions are met, the nodes execute them. Because the underlying virtual machine is more complicated, smart contracts can implement ticketing/refund systems, or voting systems, or parallel cryptocurrencies with additional bells and whistles.

WHAT'S THE CATCH?

We already touched on one of the problems associated with smart contracts on Ethereum. If you make an error in your code, the program could potentially burn through a lot of your money. There are other difficulties to contend with though. Some are technical, and some are social. We'll deal primarily with some of the social ones.

You can't always get what you want

Imagine for a moment that you have an idea for a smart contract to manage the share ownership of your company. The contract will create a limited number of shares, and will allocate them to the current owners of the company. The idea is then that share owners can trade them on the blockchain with other people, without the need for board approval or the involvement of a broker. Smart contracts aren't like web apps, or smart phone apps, where if you find a bug you can simply push out a new version as and when you please. Therefore you need to define carefully what the smart contract can and can't do. Are you going to issue more shares later? Will there be future share split? If investors come on board, are you going to have to issue a different class of shares? What happens if a shareholder accidentally transfers their shares to the wrong person, or to a totally unclaimed address – can this be fixed? Are you going to allow share-based voting on the blockchain too?

Presumably, you are not a hot-shot Solidity programmer yourself, so you will have to hire someone to write the smart contract for you. Are you capable enough to review what they produce, and ensure that what you described in English to the programmer is what was then coded up for you? Has the code been thoroughly tested, and can you be sure there isn't a subtle back-door in the contract that allows your distributed on-blockchain company to be taken over? Even if you know that your programmer is competent and trustworthy, experience has shown that when there is code, there are bugs, and they can be subtle and disastrous.

The ghost in the machine

If you are trying to move a legal contract into the realm of computer code, you're going to face a problem. The law doesn't just codify agreements, it carries with it a "spirit of the law", and we rely on judges, juries and lawyers to interpret the meaning behind the laws we create. Even patent law, which is one of the most formal and codified sectors of the legal system, still contains non-formal concepts such as "obviousness", and "inventiveness". A smart contract might be able to track a work-flow through, for example, the application for a patent, but it won't be able to determine if other patents or scientific papers constitute prior art, or if the application is for an idea that is actually novel.

Similarly, something that a person can immediately spot as a mistake (should that transaction to pay the entire company's bank balance to an unknown address really be executed?) is not going to be caught by a smart contract unless it's specifically coded for.

Your users will include hackers

If you are running your smart contracts on a private version of a blockchain, then at least you know who your users are, and when someone misbehaves you can track them down and resort to the legal system for any major infraction, or just kick them off for minor misdemeanors. But on a public blockchain like Ethereum, anyone can see your code, and potentially interact with it. The world is full of people who seem to have nothing better to do than to sit at a terminal and probe for weaknesses, write malicious code to take advantage, or even just engage in simple vandalism. When dealing with blockchains that carry crypto-tokens with real cash value, there is an added strong incentive for hackers to look for ways to steal, rather than to just engage in mischief. The more value your smart contract backs, the more hackers are going to try to bend it to their own will.

THE "DAO HACK"

DAO STANDS FOR
DISTRIBUTED AUTONOMOUS
ORGANIZATION

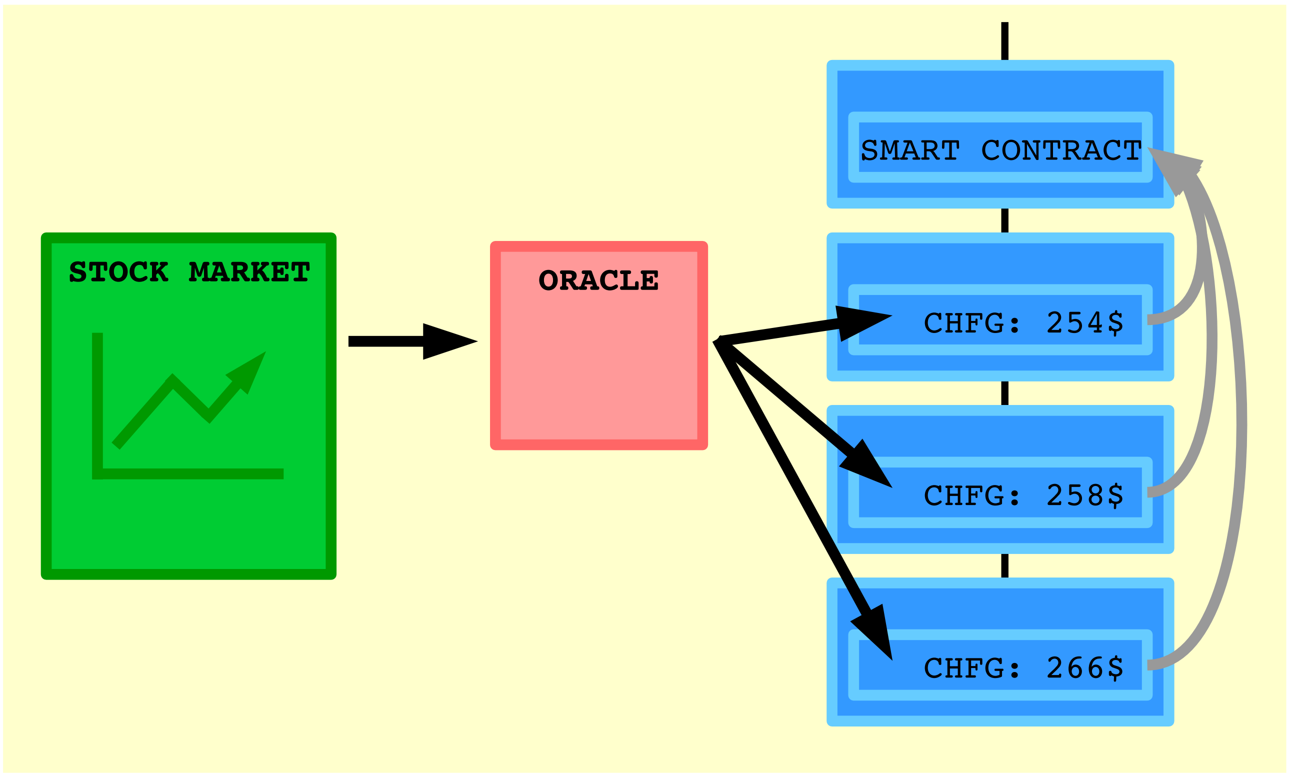
RAISED 150,000,000$ IN CROWD FUNDING

WAS MEANT TO BE A
DECENTRALIZED VENTURE CAPITAL
FUND

50,000,000$ IN ETHER TOKENS
WERE SIPHONED OFF DUE TO
A BUG IN THE SMART CONTRACT
CODE

Truth is an event

Smart contracts can only act on data that is visible on the blockchain. They don't connect out to the real world. As a result, in order to get a smart contract to act on an outside event, you need to implement an "oracle" - a conventional piece of software running on an external server that, for example, might monitor the stock price of a company, or the temperature in a given location, and then writes this data back to the blockchain in a transaction. But just because some data is on a blockchain, that doesn't make it true. If your oracle is wrong, or is hacked, the smart contract will be operating on false premises, and could end up transferring a fortune in crypto-tokens in error.



SUMMARY

So there you have it. Smart contracts are computer code that is submitted to a blockchain, where it sits until the right conditions are met, and then all the nodes on the blockchain network run it, and record the result. They can be used to provide an implementation of a workflow or payment instrument, moving virtual currency around as the situation dictates. They can even connect in to external events through outside systems called “oracles”, that forward real life data onto the blockchain.

But they carry a lot of risks. Smart contracts are difficult, or sometimes even impossible to update. They may contain subtle but disastrous bugs that can result in the loss of substantial value. And finally, they don’t use common sense in interpreting what the right action is – they just do what their code tells them to. Bitcoin uses a few simple common transaction scripts that have been run millions of times and tested over nearly a decade, and still occasional flaws or exploits surface. With more powerful smart contract systems, the opportunities may increase, but so will the risks.

In conclusion, feel free to enter the world of smart contracts, but proceed with caution!