AN INTRODUCTION TO SMART CONTRACTS AND THEIR UTILITY IN BLOCKCHAINS

Saswat Das
Int. M.Sc.
NISER Bhubaneswar

A report presented as part of the requirements of a summer project under the guidance of

Dr. Rishiraj Bhattacharyya
School of Computer Sciences
NISER, Bhubaneswar

School of Computer Sciences
National Institute of Science Education and Research
Bhubaneswar, India
12 July, 2019

NATIONAL INSTITUTE OF SCIENCE EDUCATION AND RESEARCH

Certificate

This is to certify that Mr. Saswat Das, who is currently pursuing an Integrated M.Sc. course at NISER Bhubaneswar, undertook a summer project/internship on the topic "Introduction to Blockchains" under the guidance of Dr. Rishiraj Bhattacharyya of the School of Computer Sciences, NISER Bhubaneswar, and satisfactorily completed all the requirements for the same.

Dr. Rishiraj Bhattacharyya
School of Computer Sciences
NISER, Bhubaneswar

Abstract

A brief introduction to smart contracts in the context of cryptocurrencies like Bitcoin and Ethereum, the tools that make them possible, and the ways in which they are used and can influence the future of cryptocurrencies, contracts and the internet itself.

1 Introduction

In the previous lectures we discussed different mining puzzles, the properties of some altcoins in brief, and how the peer to peer network works, how the blockchain is maintained by the participants in the same and how transactions and carried out, all the while preventing possible malicious activities by the use of incentives and precautions, and more detailed stuff. In short, we have seen how it makes for a secure and pseudonymous cryptocurrency system with the provision of making transactions online without the need of revealing your identity to a third party entity, especially in the context of blockchain ledger enabled cryptocurrencies like Bitcoin.
That being said, sure it does make for a pretty neat virtual currency, but what if it could be used in creative ways so as to enforce prewritten contracts, or to ensure the honesty of both parties in a transaction or the fairness of a transaction or even run programs and "applications" which directly or indirectly require the usage of the cryptocurrency?
As it happens, it is indeed possible to program such contracts and have the P2P network and the protocol take care of its execution and fulfilment, thereby removing the need for any intermediary like a notary and the high costs it may take to employ said intermediary, not to mention that to employ one would require you to divulge your identity to a third party. Such contracts which stay within and are executed on the blockchain are called smart contracts. And in this article we will look at how such smart contracts are used, what are the tools that make them possible, their benefits, and what this could mean for the future of all contracts, transactions, law and how things are run on the internet and daily life itself. Also we shall look at innovative ways in which transactions can be conducted for different special purposes.

2 Essential Features of Smart Contracts

Now, smart contracts need to fulfill some essential requirements, that is they need to have some essential features, to make them truly useful as a self enforcing contract on the blockchain. The contract should be costly or even impossible to breach and also, contractual clauses can be tied to the software and hardware being used for their execution (most smart contracts we will come across now are enforced with the help of software, and not as much with hardware). We will shortly see how these requirements can be realised with the help of some very ingenious ways of using the arsenal of tools available to the

user in the context of Bitcoin and Ethereum, and will gain an insight into how smart contracts are designed without going into superfluous abstract details. In addition, they should be practically impossible to breach, eavesdropped on, or having its working interfered with by malicious entities like black-hat hackers, which is achieved mainly by the use of cryptographical means, by using preventive measures or retaliatory/punitive measures (the latter being put into effect by the principals or actual law enforcement officials).

2.1 Basic Objectives of a Smart Contract

However, there are a few basic objectives that a smart contract must satisfy, according to Nick Szabo in a paper which he published in 1997, who originally proposed the concept of smart contracts.

2.1.1 Observability

This simply means that a user of a smart contract (called a *principal* in commercial law jargon, and referred to as such in Szabo's paper), can view how instances of the same smart contract is operating for other users as well as they should be able to show another user of the same smart contract how their instance of the contract is working for them. This ensures transparency and impartiality in the working of a contract, and helps pinpoint any possible errors in its working, if any. And this is important, because if a contract is not observable, it is tantamount to having a person illiterate in, say, Arabic sign a contract in Arabic, or making them sign a contract they haven't got the chance to properly even read before taking a decision to sign it, with possible clauses in it the principal doesn't even know about (hidden actions) and that can lead to a breach of privacy of the user, or leave them vulnerable to certain unwanted and unforeseen consequences as a result of said unknown clauses. (Hidden actions can be as simple as one's searches transactions, crypto or fiat currency based, being logged by the web browser and its developers).

2.1.2 Verifiability

Verifiablity of a contract refers to, roughly as Szabo put it, the ability of the principal to, when necessary, to prove to an abjudicator whether the contract was breached or was successful in its performance or the ability of the said abjudicator to find out about the same using certain means. Verifiability is important as it allows the contract to command confidence and any breaches to be addressed effectively so as to maintain that confidence. No one who breaches a smart contract should be able to walk away from it without any adverse consequences, because if people can, then the contract is next to useless and won't command confidence.

2.1.3 Privity

Control over a smart contract should only be yielded by parties that are privy to it, that is only the principals and agents who have signed on the same, and control and knowledge of the working of, inputs given to and progress of the contract should be made available only as much is necessary to a party, depending on their relationship to the contract. That being said, no third party should be able to access information about an instance of that contract as they are not privy to it, being of course an external third party (eavesdropping), or derail the operation of the contract to benefit themselves and siphon away some kind of service or resources from the contract or maliciously disrupt it. In short, in the context of smart contracts, as Szabo mentioned, third parties should lack both any control over or any knowledge of the smart contract between some parties.

3 Some Additional Features of the Bitcoin Scripting Language

A vanilla Bitcoin transaction, so to speak, typically takes input from the Unspent Transaction Outputs (UTXOs) of one or more address(es) and then specifies an output in the form of a script, in which signatures are verified (the UTXO's script and the new input's scripts are concatenated are run together to prove the ownership of said UTXO by the sender, and then the verified public key of the sender is used to verify the sender's signature on the transaction), and if the included signature is valid then the input is consumed to produce outputs to the recipient and the change is sent back to a change address specified by the sender, and in addition the difference between the value of input and that of the output, if any, is credited to the miner of the block in which the transaction is in the form of transaction fees. This is basically a scenario like in which Alice sends Bob some Bitcoins and gets the change back after paying some transaction fees. Also one or multiple input(s) can be used to produce one or multiple output(s), as we have seen in earlier lectures.
But what if additional conditions could be imposed, such that only if they are fulfilled, the said transaction will go through? Here we are entering the territory of smart contracts in Bitcoin which are quite useful but Bitcoin's scripting language being Turing incomplete and bereft of loops (to avoid potential infinite loops which can cause huge problems) and being restricted to a total of 256 commands, many of which are not accessible by the user, the types of smart contracts that can be possibly created using this language is fairly limited, but they can be created. Some of the key features that allow smart contracts to be created in Bitcoin, and some other crypocurrency systems are **multi sig** (multiple signatures feature), and **locktime** etc..

3.1 Multi Sig

The multi sig feature is one of the most essential features of Bitcoin and it aids in the creation of smart contracts in the same, or some special multiparty transactions. The basic idea behind multi sig is it requires a minimum, let's say, m number of signatures out of a possible n signatories (where $n \geq m$) to validate a transaction from an address controlled by n different public keys/co-owners.
It can be used in quite a number of interesting ways as we shall see shortly.

3.2 Lock Time

This is a parameter which is included in the metadata of a transaction, along with other parameters like the hash of the entire transaction (which helps refer to this transaction in the future), the number of inputs and outputs and the size of the transaction. Lock time is a special feature which only allows a transaction to be redeemed after a set amount of time has passed. This can also be used in creative ways to create smart contracts.

3.3 If/Else Statements

Users can utilise commands such as OP_ IF, OP_ ELSE, OP_ ENDIF to create transaction outputs with conditions built in. This feature can be used to create interesting smart contracts for applications in different contexts.

4 Smart Contracts in Bitcoin

Now, as mentioned before, smart contracts in Bitcoin can be only of a limited variety, mostly restricted to managing transactions and flow of cryptocurrency, and a few other purposes.

4.1 Multi-Sig Addresses

Suppose there is a company with n number of co-owners, and that company has its Bitcoin (BTC) funds stored in a particular address. Now of course, every co-owner has to have some access to the same. If it is achieved by the generation of a single key pair for the address that every co-owner possesses, then the funds become vulnerable to being decamped with and diverted by one of the co-owners unilaterally, in simple terms, they can run away with it, or make unilateral financial transactions using said address without the consent of their colleagues or it may so happen that if one co-owner is compromised and his secret key is stolen by a miscreant then the entire company's BTC reserve in the address can be stolen by the offender.
In one secure way to avoid this, smart contracts using Multi-Sig can be created. For a Multi-Sig transaction from a Multi-Sig to be successful, a minimum of m out of n public keys (where $n \geq m$) have to sign that

transaction for it to go through. First, n number of key pairs, each corresponding to each co-owner, are produced. Then using a special function using the value of m (minimum number of required signatures), and the public keys of the co-owners a *multi-sig address* is created. Any UTXO directed to this address must go through the m-of-n Multi-Sig process to be used in a transaction.
This can be used to strengthen the security of Bitcoins held in cold storage as well, because if one secret key corresponding to the cold storage gets stolen, the offender cannot still get access to the BTC in the cold storage and the owner(s) can retrieve the other secret keys and transfer the funds to another cold storage address and secure it again.

4.2 Escrow using Smart Contracts using Multi-Sig

Here, akin to the m-of-n Multi Sig smart contract used above, a 2-of-3 multi-sig address can be set up with the buyer, seller and a third-party unbiased arbitrator to which the buyer sends Bitcoins to. Now we term the BTCs in this address to be in escrow. If the transaction goes through without any issues, then the buyer and seller sign a transaction to the buyer's address and there is no need of the arbitrator here. But if a dispute arises then the arbitrator can rule on which of the two parties in the transaction is right and then either signs a transaction with the buyer to refund the money back to the buyer's address or signs one with the seller to have the money sent to their address.

4.3 Repeated Micro-Transactions

Let's say a buyer is a customer of a service that charges, say, per minute or per view, like a telephone, internet providing or video-streaming service. Now, paying for each call made or movie or episode watched would require numerous transactions of extremely small values. Now the problem here is that if many such microtransactions were to be made separately, then it would lead to the accumulation of a huge transaction fee cost on the buyer's end and the seller would have to worry about a lot of transaction confirmations. A neat way of resolving this would be creating a 2-of-2 multi-sig transaction with a set amount of money in the corresponding multi sig escrow address as input and the output being the latest total amount to be paid to the seller and the change being paid back to the buyer at a change address. Every time the buyer uses the seller's service, they can update the output details and sign it and leave it open for the seller to sign. Note that the transactions don't get broadcast to the P2P network yet. Also note that each updated transaction is like a double spend of the previous one, and so the the older transaction is practically voided as soon as a new one is created. Once they are done using the service or at the end of a period of time, say, a month or a year, the seller can sign the transaction too and the transaction is published on the blockchain, solving this problem.

But if the seller decides to leave the transaction hanging in this escrow address for an indefinitely long amount of time, the buyer is at a major inconvenience as they cannot use any of the money they have placed in escrow in the Multi-Sig address, including the change. So here as part of their agreement, both the parties can sign a transaction which refunds all the money back to the buyer with a lock time value, at the completion of a particular time period (when the lock time lapses), if the seller has not yet signed the above-mentioned transaction, then the buyer can publish this refund transaction and get all of the money in escrow transferred back to their address.

4.4 Betting on a Match using a multi-sig address

Let's say A and B bet on the outcome of a coin flip (which can be reasonably simulated using a good random number generator) or a match, then they can settle it using BTC transactions. Let's say they have bet on a cricket or football match in this manner and as the match progresses it becomes apparent that it is not B's day, then they can just pull the plug on the bet and make a run for it by refusing to sign the corresponding transaction for losing this wager. This can be taken care of by using smart contracts. They can bet money using a 2-of-2 or 2-of-3 multi-sig address (if they wish to employ a third party arbitrator to resolve any possible disputes) and sending money to it and using it as a "pot". This way the loser cannot just run away with their share of the pot just like that. If they stubbornly refuse to sign any transaction to the winner and make things difficult, then if employed, the arbitrator can step in and resolve the situation, or a timed commitment could be used, as mentioned in the next example.

4.5 A Betting Game using Timed Commitments

Could we make a kind of application/contract to run a betting game? Let us look at this particular game encoded as a contract, which is a variation of the one mentioned in the book "Bitcoin and Cryptocurrency Technologies". Let's say A, and B decide to play a betting game in which each of them randomly pick a large number and hash it to commit to it using a hash function, optionally concatenating it with a nonce then hashing it, and then when done they can report the commits/hashed values. Then they can reveal the nonce (if used) and number and then add both of those numbers a, b (corresponding to each of A, B respectively) and then compute $(a + b)\%2$ and then depending which of the three possible outputs $(0, 1)$ we get, A or B can be rewarded the pot of wagered money/BTCs using if/else statements in the transaction script. Then again, it requires A and B to be honest to the scheme and be willing to sign the final transaction to pay the winner. Say A has revealed his/her number and B knows that he/she are going to lose, he/she might go offline and not reveal their number. To prevent this we can use a timed commitment scheme.

That is we can use OP_IF, OP_ELSE and OP_ENDIF (which ends an if/else block in a transaction's script) to create conditions with the use of lock time. A could sign a multi sig transaction with B using a bond in a multi sig address(not part of the wager) consisting of some amount of BTCs that pays the bond to B after a certain lock time t elapses. Then they can create a transaction without any lock time that pays the bond back to A either if

1. A and B both sign it or,
2. only A signs it to redeem this transaction if and only if A reveals his/her secret number and verifies the commitment as well within the script.

So it is in A's best interest to reveal and verify his/her secret number before the lock time t expires and publish the transaction which pays the bond back to him/her. If A does not reveal his/her secret then they lose the bond, and how willing they are to lose said bond depends on the size of the bond vis-à-vis the amount they have bet.

5 A Brief Introduction to Ethereum

In our previous discussion about alternative mining puzzles we talked about how some altcoins are mined and some of the special features of those altcoins. Ethereum, at first glance, could be taken for, unerroneously, as one such altcoin. But Ethereum has certain special features that push its utility way beyond the realm of just being a blockchain enabled cryptocurrency used for transactions.
Some interesting and salient features of Ethereum is as follows,

1. It uses a scripting language called *Solidity* which is Turing complete and hence supports loops without any limit on the number of possible instructions, which Bitcoin's stack based scripting language and that of many others don't have, other turing complete languages like Serpent and Mutan are supported, and as of late even C++, Ruby and Python are supported by the Ethereum Virtual Machine (Ethereum's runtime environment, discussed below), and they can also be used to write smart contracts on Ethereum;
2. It uses an account based system as opposed to the UTXO (Unspent Transaction Output) system used in Bitcoin to track the account balance of each user, which requires special data structures (based primarily on the use of a modified Merkle tree called a Merkle-Patricia tree), but is more efficient and faster to use;
3. Each block ideally stores the details of each account's state and account state history, all of the smart contracts that exist (their latest versions), and transactions, and any data on the blockchain (the method using which this is achieve is out of the scope of our current discussion);

4. Everything that is done on Ethereum is done by the use of contracts, the execution of each contract uses up some amount of ether which is used to compensate for the amount of computing power used for its execution;

5. Every transaction or smart contract is validated by every node on the network;

6. Every instance of a contract is read by and executed by an Ethereum Virtual Machine (EVM), which is Ethereum's runtime environment;

7. Currently uses Proof of Work using a memory hard puzzle called Ethash using a consensus protocol which resembles the Satoshi consensus protocol but might switch to a Proof of Stake mechanism as early as this year (which would make it ASIC resistant and more decentralised);

8. A new block is published every 12 seconds with a block reward of 5 ether, which facilitates faster confirmations and execution of contracts;

9. A new improved protocol for Ethereum, called ETH 2.0 (with the aforementioned Proof of Stake update), will be implemented atop the existing Ethereum protocol layer, to improve Ethereum's scalability and efficiency while retaining its security features.

Note: In Ethereum's account-based transactions, the sender (say, X) basically signs a contract which has the message "Pay Y an amount of n ether from X's balance". Now the issue here is that the receiver. Y, could repeatedly broadcast this message to the P2P network which would practically deplete X's balance. To avoid this, each account has a parameter (transaction number) which keeps track of how many transactions have been completed and each transaction from that account is issued using the current transaction number. Once this transaction is completed, the transaction number of the sender X would increment by 1, hence preventing this issue. This issue is not seen in UTXO based systems like Bitcoin of course, as they operate by consuming the entirety of previous unspent transaction outputs to create new outputs.

5.1 Potential Uses of Ethereum

Ethereum, by virtue of EVM supporting Turing complete languages, like the primarily used one called *Solidity* (which is inspired from C++, Python etc. and can support inheritance, libraries and complex user defined types) and the EVM's capability to work using other Turing complete programming languages can be used to create increasingly complex smart contracts, some of which can practically emulate any altcoin. For example, a smart contract can be be written (just like a program) to consume some units of Ether and produce a Cunningham prime series of a desired length, as is the case in PrimeCoin. This language can be used to create complex programs within contracts that can run like some applications and do complex tasks such as store notes on the blockchain, or be used to play games or even host a photo sharing service.

As described by the creator(s) of Ethereum, "Ethereum is a global, open-source platform for decentralized applications." Smart contracts here are alternatively known as "dapps" (decentralised apps).
To bring into perspective how important such applications which run in a dencentralised manner on the blockchain are, we could pay attention to the fact that when the Internet was originally designed, it was ideally meant to be a decentralised system with each computer on the system contributing more or less equally to it without the concentration of data in the hands of some entities, which is clearly not the case. Ethereum aims to realise the idea of such a "World Computer" fueled by ether which is primarily used to pay for the use of computing power of this "World Computer" (which is nothing but the computing power of the nodes involved in the P2P network).

6 Smart Contracts in Ethereum

Smart contracts in Ethereum can be used for various practical purposes. Again, smart contracts can be viewed as (as described by Nick Szabo in a paper published in 1997) vending machines which can render a customer some service without the need of any middleman per se and they define the regulations and possible penalties associated with it, as with any other traditional contract and enforce said obligations. In short, a proper smart contract is a self-enforcing agreement, which is possible especially in the context of Ethereum. Every smart contract's execution requires a small payment in Ether to get done, in the absence of which the contract will simply be "thrown", i.e. it will go unexecuted.

6.1 Financial Applications

Smart contracts on Ethereum can be written to facilitate financial operations like transactions, lending, borrowing, investment in terms of your cryptocurrency and digital assets in a convenient and secure manner by the use of ether without tht involvement of fiat currencies or physical assets.

6.2 Voting

By virtue of the account system and the input voting data being stored in a tamper evident blockchain ledger, a smart contract to record signed votes from voters registered on the voting list with their Ethereum accounts can be used. Signatures are unforgeable and any tampering to the voting data in the blockchain cannot be done in an undetected manner, and hacking the entire set of voting data would require an unimaginable amount of computing power. This could open avenues for online and remote voting without any fear of impersonation (the owner of an account and the corresponding key pair can only vote from that account) and can increase voter "turnout" in the way that voting would be made more convenient and secure than ever.

6.3 Purchase of Goods/Assets

Smart contracts can be written, akin to the 2-of-3 multi-sig systems with the buyer, seller and a third party arbitrator to ensure the fairness of the transaction we discussed before, to ensure that a payment will only go through after the goods or assets have been properly transferred to the buyer with the completion of requisite formalities and after production of proof of the same. For instance, as it happens Barclay's Corporate Bank actively uses smart contracts to keep track of change of ownership of assets or conduct movement of money.

6.4 Maintaining Accounts and Ledgers

Smart contracts can be written to collect and conduct required computations and calculations on input raw data and store them in a tamper evident blockchain ledger, which is secure and easy to access by anyone when required. Also any algorithms for complex calculations can be coded into the contract which can then compute on the data and store the processed information in an organised manner as specified in the contract on a blockchain, and the contract can further be designed to check for discrepancies in the ledger and accounts and make the entire process of maintaining records hassle free and error free.

6.5 Playing Games like Chess with Bets

The rules of chess can be coded onto a smart contract using Solidity, including a provision for possible bets and wagers in terms of Ether and with the inclusion of a time limit for each move so that a player cannot leave the match midway without being penalised and the betted money being awarded to the opponent for the player forfeiting the match, and a good random number generator which will help determine which player gets to play with white pieces and hence go first, and the rules for a checkmate, stalemate or draw being coded onto the contract, with rules on how to disburse the money in the "pot" in the case of each outcome. This requires no central authority and this game and its outcome remains restricted to these two players solely. Similarly, more online games can be created using programming languages in the form of smart contracts involving any number of players and possibly with betting.

6.6 Use in Supply Chains

Smart contracts can be used to make payments when the required goods are received by the customer with the proper procedures and formalities being done. Once the payment is made, the sale can be recorded onto a blockchain and it could intimate the manufacturer/producer about it fairly quickly, and with Ethereum's low block time, it is extremely fast as compared to paperwork based confirmation which changes hands many times from the courier right to the manufacturer, taking a lot of time. Assured of this particular sale, the

manufacturer can start producing a new unit of the same product if required. This system is already in use by companies like UPS.

6.7 Insurance

Smart contracts could be used to handle insurance premium collection, record keeping and redemption based on the insurance agreement and the conditions present in it in an impartial manner and thus can prevent disputes or any unfair practices on the company's end to deny redemption of the insurance amount.

6.8 Web Streaming

Artists, record companies and film companies can directly store their content on the Ethereum network (in a complex storage scheme where the files may be stored in pieces scattered over the network) and stream it to the public with the help of a smart contract that does all this and mimics a streaming service like that provided by YouTube, Netflix or Spotify, but without the involvement of a middle man and any payments that have to be made to the creators of any content can be specified in the smart contract, upon completion of which a user can access said content for a specified period of time and with certain terms and conditions. A user can also tip the creators in the case of a free-to-view streaming service. Plus the creators have to pay a certain amount of ether for using the service/smart contract to upload their content onto the blockchain. Services like D-Stream currently operate using this model on the Ethereum network.

6.9 Store Membership Records

Membership formalities could be done and membership fees, if any, could be paid using smart contracts which would then log the information onto a blockchain which would be convenient for recording details and records about members. This can also be viewed as a name registry that can be stored on the blockchain on nodes on the Ethereum P2P network, and this would, in essence, mimic the properties of Namecoin, another altcoin.

6.10 Crowdfunding

Instead of using services involving a third party like GoFundMe or Kickstarter for the purpose of crowdfunding, one could design a smart contract using Solidity which would stay on the Ethereum blockchain and would work as follows. The contract would allow setting of a funding target and people can pay money onto an escrow account handled by this smart contract without the involvement of any third party and when the funding goal is achieved then the funds could be directed to the intended recipients. If the goal is not achieved after a considerable amount of time (using lock time), then the transaction

forwarding the money to the recipients gets cancelled and the money gets returned to the recipients in a refund transaction.

6.11 Daily Spend Limits

If two users have a joint Ethereum account (like a 1 of 2 multi sig account), then a smart contract can be imposed on it such that each user can spend only a very small percentage of the existing Ethereum account balance in terms of Ether per day, thus regulating how much money is spent in a day while making sure that a user cannot make off with the entire balance at a go in a certain situation (say, if things go awry between the two users).
Note: This is not possible in the context of Bitcoin as there is no record of Bitcoin balance per se on the blockchain for a given address and each UTXO is treated like a separate one, unless they were spent in a transaction back to the same user in a transaction that takes all those UTXOs and turns them into one large combined UTXO, which again has its own issues and implications (If the change address is wrongly specified even with a small margin of error during a transaction with a large UTXO, then the user stands to lose a lot of BTC), which are not present in Ethereum's account based model.

7 Conclusion and Remarks

Clearly we can see that smart contracts can be made for virtually any purpose using Solidity and the Ethereum network and can be executed in a decentralised and convenient manner, and complex smart contracts mimicking several web services can be made. This could in the long run not only make for a great cryptocurrency system but it could in a way help realise the vision of a decentralised Internet and eliminate many middlemen who could be agents of centralisation and avoid the concentration of a lot of data and thus control over the internet in the hands of a few, and could lead to a more transparent Internet, should this paradigm be adopted; but this should be taken bearing in mind that sophisticated contract and blockchain-based data splitting and storage mechanisms will have to be used (merging some principles of PermaCoin with Ethereum might be useful). Also this would affect lawyers and jurisdiction etc. (as smart contracts are self enforcing) and laws may have to be made for what kinds of smart contracts are allowed and what are not, but that is something only time will tell. The least we can expect is some sort of paradigm change in the way we conduct our transactions and agreements in the years to come as smart contracts are getting increasingly adopted by people and organisations.
This was a brief look at smart contracts, their creation, and their utility; but this is but the tip of the iceberg as smart contracts keep on getting increasingly complex by the day and research is being carried out on the same and on their possible impact and on how they can be used in different ways.

References

1. Narayanan, Bonneau, Felten, Miller and Goldfeder. *Bitcoin and Cryptocurrency Technologies*. Princeton: Princeton University Press, 2016.

2. Rosic, Ameer (2016). "Smart Contracts: The Blockchain Technology that will replace Lawyers". Blockgeeks Inc (online).

3. Szabo, Nick (September 1997). "Formalizing and Securing Relationships on Public Networks". First Monday.

4. Multiple authors (2017). "A Beginner's Guide to Blockchain Technology". Coindesk (online).

5. Solidity. Stiftung Ethereum. https://solidity.readthedocs.io/en/latest/

6. Simply Explained - Savjee. "Smart Contracts - Simply Explained." YouTube. 20 Nov, 2017. https://www.youtube.com/watch?v=ZE2HxTmxfrI/

7. D-Stream, "D-Stream - Blockchain based decentralized video sharing platform". 17 May, 2019. Retrieved from https://labs.imaginea.com/d-stream-blockchain-based-decentralized-video-sharing-platform/

8. https://www.ethereum.org/. Ethereum.

9. En.wikipedia.org. (2019). "Ethereum". (online) Retrieved from https://en.wikipedia.org/wiki/Ethereum.

10. Saini, Vaibhav (29 July, 2018). "Getting Deep Into Ethereum: How Data Is Stored In Ethereum". Hackernoon (online). Retrieved from https://hackernoon.com/getting-deep-into-ethereum-how-data-is-stored-in-ethereum-e3f669d96033