Security Assurance for Smart Contracts

Group 8 Project Report - CSE 543 - Fall 2019

List of group members, including group leader and deputy leader:

Michael Sneberger	Leader
Sujay Gopanahalli Devaraj	Deputy Leader
Harsha Vardhan Kaki	
Sarthak Shetty	
Harsh Lalwani	
Varun Krishna	
Siddhartha Sriram Vinjam	
Poonam Nair	

Summary

Group 8 undertook an investigation of what is a blockchain, what kinds of blockchain platforms are available and which support smart contracts, what is a smart contract, and how parties to a smart contract on top of a blockchain platform can be assured of their veracity and security, and what are the security issues in smart contracts and ways to handle them.

Introduction:

A blockchain is an open distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way. Transactions between parties that are meant to be memorialized in a blockchain are grouped together to form a block which is then subject to some consensus algorithm whereby nodes in the network approve adding the block to the chain. Once this occurs the block become immutable.

Blockchain platforms enable the advancement of blockchain-based applications. They can be permissioned or permissionless. Ethereum, Hyperledger, R3, Ripple, and EOS are some names that have built blockchain frameworks, letting people build and host applications on the blockchain

Smart contracts inherit properties of underlying blockchains which include an immutable record of data, and the ability to mitigate single points of failure. Smart contracts can also interact with each other via calls. Unlike traditional paper contracts that rely on middlemen and third-party intermediaries for execution, smart contracts automate contractual procedures, minimize interactions between parties, and reduce administration costs

A digital signature is a digital representation that mimics the process of manually signing a contract with a signature in ink. Given that "wet" signatures in ink can be forged, or the contract to which a "wet" signature is affixed can be altered without the signer knowing, an unalterable digital signature provides a more secure way to show that a party has indicated their agreement to adhere to the terms of a particular contract. More specifically, while a "wet" signature on a paper document cannot meet the criteria for an effective signature, a digital signature scheme can meet those criteria.

Formal modeling approaches to the verification of smart contract behavior can be applied to concrete smart contract examples and analysis of breaches with a statistical model checking approach can be achieved.

Full-service Solidity language on the Ethereum blockchain platform allows developers to program their own smart contracts. Smart contracts have been used for many applications such as: Digital Identity; Banking; Tax Records; Insurance; Real Estate and Land Titles Recording; Supply Chain; IoT(Internet of Things); Authorship and Intellectual Property Rights; Life Science and Health Care; and Gaming and Gambling.

Due to the inherent structure of the blockchain itself or due to the coding practices followed by the developers writing the code that facilitates smart contracts, developer can face multiple problems and issues while developing smart contracts for the Ethereum blockchain platform.

Given that smart contracts can and do represent significant value to the parties, vulnerabilities in smart contracts present serious issues, and require fixing at early onset for many reasons, not the least of

which that bugs in smart contracts, once they are developed, cannot be fixed because smart contracts are immutable in nature. Thankfully, there are multiple avenues for assuring the integrity of smart contracts.

Ultimately, smart contracts utilizing the structure of a blockchain platform are already in use, and will surely provide the efficient future of contracting and tracking in many areas of human endeavor, and the World will rely on computer scientists to design and facilitate these contracts in a secure manner.

Motivation:

Group 8's motivation for its report is that Group Members understand having an understanding of smart contracts and their underlying blockchain technology will not only have an impact on their professional lives but on their personal lives as well as more and more contracts move to a virtual, "smart" form. Thus it is important that Group Members gain an understanding on blockchain, smart contracts, and information assurance concerns related to smart contracts on top of a blockchain.

Background:

Blockchain technology is growing in prevalence in the realm of computer science, and it appears that smart contracts will be one of the early ways that blockchain technology will impact the general public. Now that more and more commerce is being undertaken online, between people and organizations in disparate locations, over publicly accessible channels, the antiquated process of affixing "wet" signatures in ink to paper contracts can be accomplished more efficiently with smart contracts, and smart contracts are often facilitated by an underlying blockchain technology platform. Click-through contracts online for things such as terms of use and software licensing function well, but for more peculiar contracting, the ability to write specific contract terms by way of a complete language sitting atop a blockchain protocol is necessary.

Goals and Scope:

The goals of Group 8 regarding the report is to generate a document, the scope of which is such that would allow a sophisticated reader to gain a greater understanding of the assurance of blockchain and smart contracts, and show why smart contracts can be relied upon as a secure means of memorializing a contract's terms and facilitating the parties' agreement to those terms.

How the Report is Formatted:

Group 8 took the rough outline of a how to take a survey of blockchain/smart contract issues from a 2017 article by Maher Alharby and Aad van Moorsel. [1] With some adjustment, Group 8's review of smart contracts on top of a blockchain pattern is structured as follows:

Section	Description and Author	Pages
1	**Description of Blockchain:** this section was written by Group Member Harsha Vardhan Kaki and provides the reader with an overview of how a blockchain is structured and functions.	8 - 11
2	**Blockchain and Software Platforms that are Used for Smart Contracts:** this section was written by Group Member Poonam Nair and provides the reader with an overview of the various blockchain platforms available, which support smart contract creation, and what language can be used to create such smart contracts	12 - 16
3	**Description of a Smart Contract:** this section was written by Group Member Sarthak Shetty and provides a general description of a smart contract.	17 - 19
4	**Description of Digital Signatures:** this section was written by Group Leader Michael Sneberger and provides an overview of the history of asymmetric cryptography and digital signatures including what technology is used to facilitate digital signature in the Ethereum blockchain platform.	20 - 22
5	**Description of Formal Verification Protocols:** this section was written by Group Member Varun Krishna and outlines a formalized modeling approach to verify the behavior of a smart contract in its execution environment.	23 - 27

Finally, Group Leader Michael Sneberger, along with help from Sarthak Shetty and all Group Members, edited the Report so that the List of References, and citations to those references was in the proper format, and so that the Report reads as one cohesive document.

Responsibilities of Each Group Member:

MEMBER	RESPONSIBILITY/TASK ASSIGNMENT
Michael Sneberger	- Assure delivery of Project documents as due - Review applicable source material and assess the value - Study and cover digital signatures for the report
Sujay Gopanahalli Devaraj	- Assure delivery of weekly meeting minutes - Review applicable source material and assess the value - Study and cover issues facing developers writing smart contracts for the report
Harsha Vardhan Kaki	- Review applicable source material and assess the value - Study and describe blockchain for the report
Sarthak Shetty	- Review applicable source material and assess the value - Study and describe smart contracts for the report
Harsh Lalwani	- Review applicable source material and assess the value - Study and evaluate current/future ways Ethereum Smart contracts are made secure for the report.
Varun Krishna	- Review applicable source material and assess the value - Study and describe formal verification protocols for smart contracts for the report
Siddhartha Sriram Vinjam	- Review applicable source material and assess the value - Study and describe applications that lend themselves to smart contracts for the report
Poonam Nair	- Review applicable source material and assess the value - Study and describe blockchain and software platforms that are used for writing smart contracts for the report

1. DESCRIPTION of BLOCKCHAIN:

A blockchain is an open distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way [2]. Open means “accessible to all” and public ledger means history of transactions which is available to everyone in the network i.e., all the nodes have local copies of all the transactions. Permanent way represents that the transactions registered once cannot be erased and anyone cannot tamper them.

Centralised Vs Decentralised Vs Distributed:

When a single point of source holds the total network and controls the network, then it is Centralised network. Decentralised networks means multiple nodes handling the data and equal powers of network are shared by all these multiple nodes.In distributed network there is not point of administrator or owner and all the nodes in distributed network execute the jobs equally. Blockchain can be referred to as decentralised and distributed based on the context of its usage.

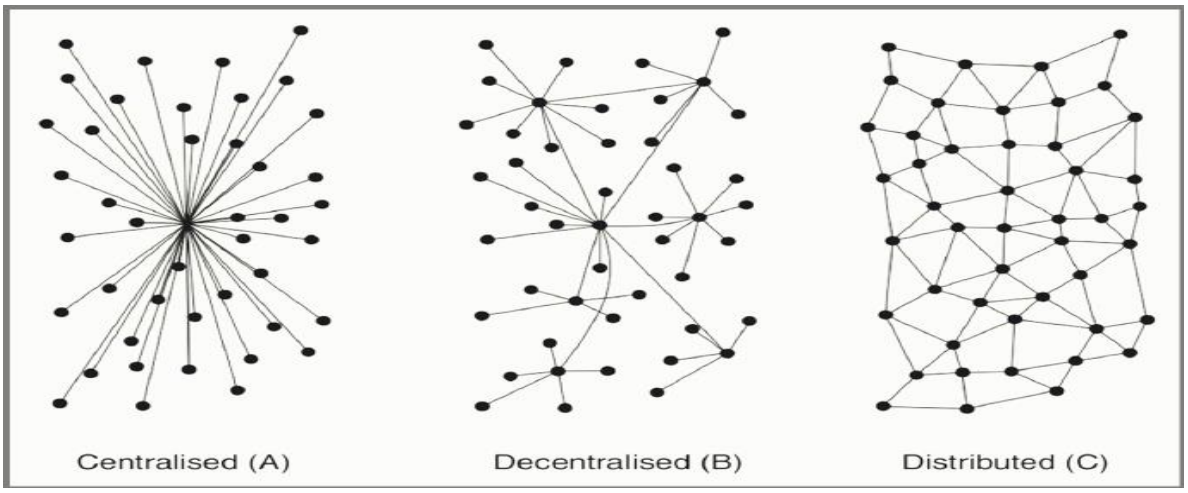


Diagram of the various ways to organize a network. [3]

Structure of a Blockchain:

The base of blockchain is merkle tree structure and hash associated with it. Merkle tree basically stores all the hash values of the appropriate child nodes.These hash values are propagated through out all the nodes till leaf nodes. We need to consider the merkle tree using bottom up approach. Every node in the tree holds the hash value calculated from it’s left child and right child. Let the current node be N and left child of N is NL and right child is NR, then hash of N can be calculated as follows,

Hash(N) = Hash(NL) + Hash(NR)

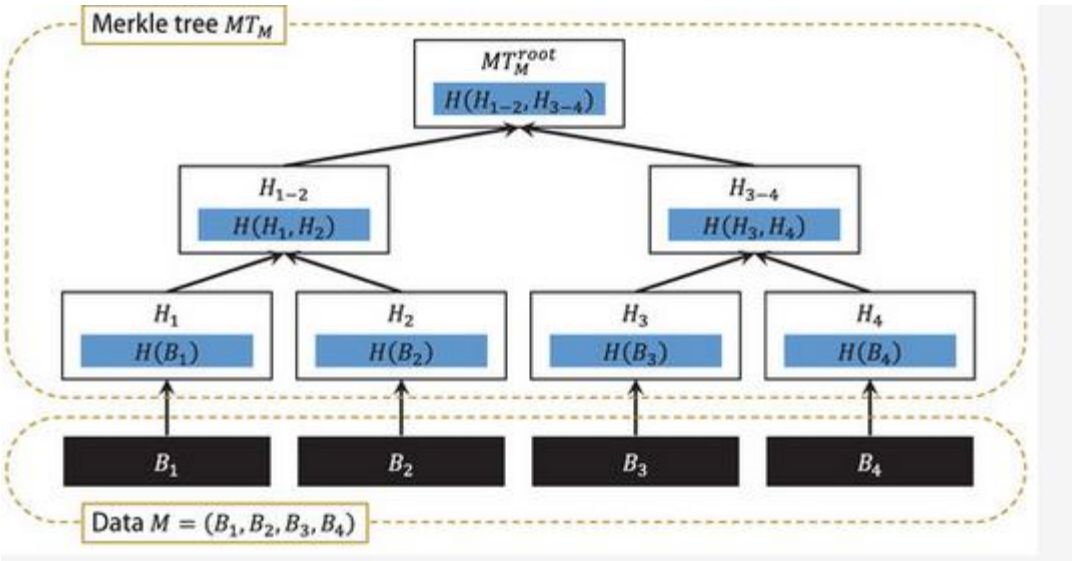


Diagram showing a Merkle Tree structure. [2]

So all the blocks are arranged in a chain in a merkle tree structure. Each block has two parts called block head and block body. Block head has four significant parts such as

- Nonce
- Time stamp
- Hash of previous block
- Merkle Root hash

And block body contains the records of the transactions stored in that block. If any transaction is tampered then there will be hash difference will be identified very easily. So blockchain's structure is considered as the World's prime structure to record data and store them securely.

Working of a Blockchain:

Working of blockchain refers to how the transactions are grouped together to form a block and how blocks are chained to form a blockchain. To do this work, there are some set of consensus algorithms which nodes in the network should follow to formulate their work. We can broadly classify these algorithms into:

- Proof of Work (PoW)
- Proof of Stake (PoS)

continued next page . . .

Proof of Work :

It is the basic and original consensus algorithm in blockchain. When a number of transactions are done in a network, all these transactions should be combined together to form a block. To combine these transactions, first validation of blocks is necessary and this validation should be done by any of the nodes in that network. The node which does this validation is called Miner node and the process of validation and grouping as block is called Mining. Not everyone can do mining, the person who wants to perform mining must solve a mathematical puzzle handling transactions and the output of this mathematical puzzle is called hash. Mathematical puzzle can be of various types such as

- Hash function
- Integer factorisation
- Guided tour puzzle protocol

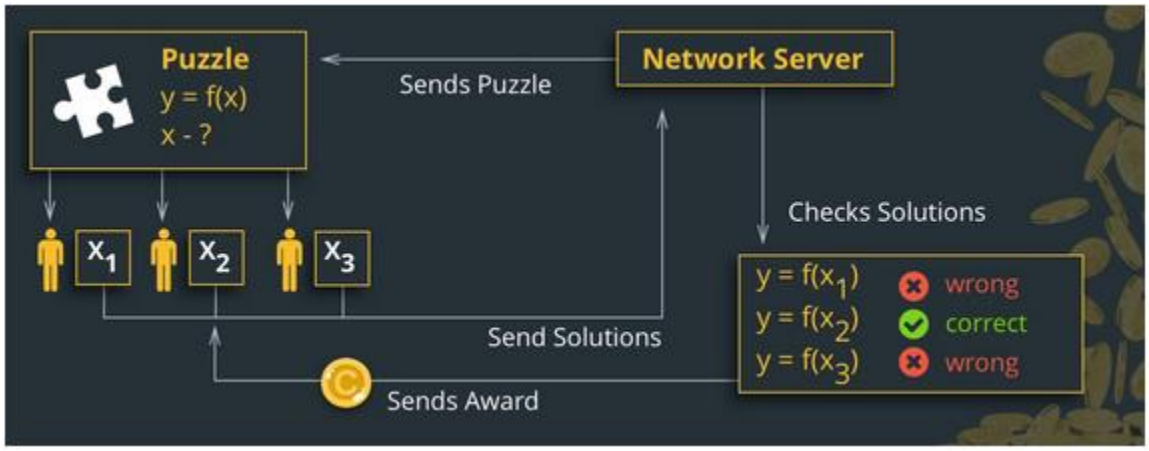


Diagram showing the flow of the puzzle solving function. [4]

The miner who solves this mathematical puzzle correctly can do mining process and he gets some reward for this. PoW algorithm makes sure that all the nodes in the network get an equal opportunity of mining so as to maintain distributed nature of blockchain. Flaw in Pow is popularly known as 51% attack in which some nodes (miners) form as a group to solve all the challenges and if that group exceeds by 50% then that group takes handover of that group and which results in security threats again.

continued next page . . .

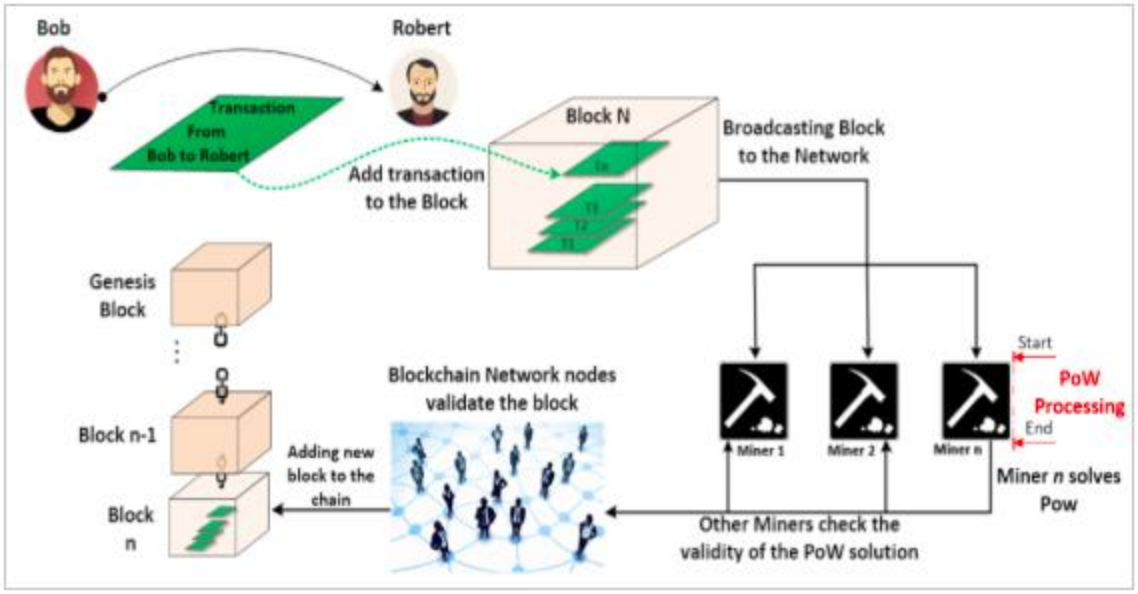


Diagram showing the flow of the mining function. [5]

Proof of Stake:

Even though Pow algorithm can be used undoubtedly for mining process, 51% attack can disturb the distributed nature of blockchain. So Pos algorithm allocates miners in such a way that only the nodes which have done a minimum number of transactions only can perform mining process. The backbone of Pos is Pow and the only difference between the two is throwing mathematical puzzle to nodes.

In conclusion, though blockchain is a robust data structure to store and transfer data such as financial transactions, there are certain flaws in the basic workings of consensus algorithms such as Proof of Work and Proof of Stake. As we showed Proof of Stake algorithm can be told as better version of Proof of Work algorithm but there are also drawbacks in Pos also. In 2016, Intel came up with a more robust algorithms known as Proof of Elapsed Time (PoET) where there are restrictions for nodes to join the network. At present, PoET is more reliable consensus algorithm available for a permissioned blockchain (2.0).

2. BLOCKCHAIN and SOFTWARE PLATFORMS that are used for SMART CONTRACTS

With the growing need for blockchain, every organization is starting to understand the capability of this technology. Originally, the blockchain brought interruption in the financial sector whereas now its benefits have been explored across diverse industries. Since the organizations have commenced analyzing the potential of blockchain by building blockchain applications, the demand for the blockchain development platform has increased greatly, leading to the development of multiple platforms.

Smart contracts are essential elements in a blockchain. Smart contracts are of great interest to businesses just like blockchains. It settles the concern of distrust among business partners. This spares time, limits irrelevant expenditures, and boosts transparency. It eliminates human participation in transactions since everything is completed via prescribed program code.

Solidity is one of the main programming languages used for building smart contracts on Ethereum blockchain platform. Solidity is a high-level programming language which is similar to C++, JavaScript and Python. These are vested with the duty of saving all programming logic that transacts within the blockchain. It also supports libraries, inheritance and elements which are statically typed and has the potential to build blockchain applications which can promote industrial strength. [6]

What is a blockchain platform?

Blockchain platforms enable the advancement of blockchain-based applications. They can be permissioned or permissionless. Ethereum, Hyperledger, R3, Ripple, and EOS are some names that have built blockchain frameworks, letting people build and host applications on the blockchain. [7]

1. Ethereum

Ethereum was developed in late 2013 by Vitalk Buterin. It is open sourced and blockchain based distributed computing platform. Ethereum runs smart contract on a custom built blockchain. Ethereum Virtual Machine (EVM) offers run-time environment to smart contracts which are programs that manage the performance of accounts that are inside the Ethereum state. Each node inside the network need to execute an EVM implementation. Solidity is an object oriented programming language which is used to code smart contracts on the Ethereum platform. One can make contracts for purposes like blind auctions, crowdfunding, voting and multi signature. It compiles into a bytecode format that is understood by the Ethereum Virtual machine (EVM). [7]

Although Ethereum has been adopted by organizations extensively, it is necessary to know that Ethereum is a public blockchain platform which has been built for restrained access versus mass consumption. It is also a Proof of Work (PoW) based platform that is relatively slower in terms of speed.

The native cryptocurrency of Ethereum is Ether. It is used for fueling the Ethereum ecosystem [7]. To develop applications, the developers need to pay in terms of Ether in order to execute transactions and run apps on the Ethereum network.

2. Hyperledger Fabric

Hyperledger Fabric comes under one of the projects of Hyperledger which is intended to develop blockchain based solutions and applications by utilizing a modular architecture. The network designers can plug in their preferred components like consensus and membership services which distinguishes it from other blockchain platforms. [7]

The framework for Hyperledger fabric is designed in a way to support permissioned networks, helping in enabling known identities to take part in a system. Authorization needs to be given to the participants within this network. The participants should also have the credibility to be a part of the blockchain.

3. Hyperledger Sawtooth

Hyperledger Sawtooth was released by the Linux Foundation and contributed by IBM and Digital Asset. It is a modular platform which is enterprise grade and is designed to create, deploy, and execute distributed ledgers that facilitate digital records to be managed without a central authority.

Sawtooth's PoET (Proof of Elapsed Time) consensus mechanism allows Hyperledger Sawtooth to consolidate with hardware security solutions also noted as "trusted execution environments". One of its applications is Intel's Xeon processor.

This is Hyperledger's second open-source blockchain platform to consolidate with an enterprise-ready 1.0 version. The deployment of 1.0 software solutions is necessary for developers as it shows the maintainers are dedicated to the core features. [7] Thus, it gives blockchain organizations confidence in whatever they develop as it won't crash due to further upgrades in the future.

4. Ripple

Ripple was discovered in 2012. Itis designed to join banks, digital asset exchanges, payment providers and corporate through a blockchain network called RippleNet without any chargebacks. It also enables payments globally via digital asset called "XRP or Ripple," which is one of the prominent cryptocurrencies like Bitcoin and Ether. [7]

Ripple is developed on advanced blockchain technology called XRP which is further scalable and faster than other blockchain platforms. It uses probabilistic voting to reach the consensus between nodes.

The potential of this blockchain platform is being tested by big companies like American Express, Santander, SBI Holdings, MoneyGram International and Deloitte and these organizations are planning to consolidate it to make the existing payment processes faster and secure.

5. Stellar

Stellar is similar to Ripple in terms of dealing with exchanges between cryptocurrencies. It is a distributed blockchain platform that is utilized to facilitate cross asset transfers of value. It is also known as a platform that connects payment systems, banks and people. Stellar network can be used to build smart devices, banking tools and mobile wallets. [7] Stellar smart contracts are not Turing complete unlike Ethereum.

6. Cardano

Cardano is a smart contract platform like Ethereum. Through its layered architecture, it offers security and scalability. It aims at increasing its scalability through its Ouroboros proof of stake consensus mechanism. [8]

One has to use Plutus, that is based on Haskell, the language used to code Cardano in order to code smart contracts. Plutus and Haskell are functional programming languages unlike C++ and most of the traditional languages which are Imperative programming languages.

7. Neo

Neo was founded by Erik Zhang and Da Hongfei who founded “Onchain”, a blockchain R&D company. It is also known as the “Ethereum of China”. It was created to develop scalable decentralized applications. Neo token is the base asset of Neo blockchain which is used to create GAS tokens that can be utilized to pay transaction fees in order to execute applications on the network.

Neo’s main aim was to develop a smart contract platform that has all the benefits of an Ethereum Virtual Machine, without weakening their developers with language barriers. The developers need to understand and learn solidity to code smart contracts in Ethereum, whereas in Neo, developers need not learn a new programming language as they can incorporate using Java, C# or other languages to build smart contracts. [7]

8. EOS

Block.one founded EOS which is a blockchain platform that was released as an open-source software in June 2018. EOS is devised for building Decentralized applications. One billion tokens (ERC-20) were distributed to establish wide range of distribution of their cryptocurrency and enable any user to operate EOS blockchain after its release. [7]

The main aim of this platform is to provide decentralized storage of enterprise solutions, decentralized application's hosting, and smart contract capability, solving the scalability issues of Ethereum and Bitcoin. It also gets rid of the fees required for all users, which means no one has to pay to utilize the benefits of a dApp (Decentralized applications) based on EOS.

EOS achieves consensus by applying multi-threading and a delegated proof-of-stake algorithm. EOS have their own community forum called EOS Forum which helps investors and developers to discuss about the platform.

9. Eris

Eris is a smart contract application platform. It is a free software which lets anyone to develop their own low cost, secure application using blockchain and smart contract technology. Its main aspect is its capabilities-based permissions, which describes who can build smart contracts, transact on the network and authenticate transactions. Eris is also an adaptive platform which uses Docker to enable a range of components to be transferred in and out with other compatible components.

Since Eris uses Ethereum VM, developers need to know Serpent or Solidity which are used in Ethereum VM. Another requirement is that the developers need to identify their own key signing daemon thus adding the requirement of having knowledge in this specific domain. [9]

It can be a feasible platform to work on if the developer can make plain applications using recommended components. But it would involve a lot of work for those developers who prefer using other components or make components on their own.

In conclusion, as blockchain is pacing at a fast rate of innovation, new platforms are beginning to develop along with added features and new releases. There is plethora of blockchain platforms which are accessible across the world, organizations have to analyze the right platforms to develop highly scalable

applications. As mentioned above, there are a variety of platforms and their specific features that make them unique. There is not a single platform that meets all the needs for now. Different organizations have different requirements and they have to determine the platform that best fits the functionalities required for their Decentralized applications (dApp).

3. DESCRIPTION OF A SMART CONTRACT

The history of smart contracts can be traced back to the 1990s when Wei Dai, a computer engineer created a post on anonymous credits, which described an anonymous loan scheme with redeemable bonds and lump-sum taxes to be collected at maturity. Szabo et al. [10] later discussed the potential form of smart contracts and proposed to use cryptographic mechanisms to enhance security. Nowadays, with the development of blockchain technology, smart contracts are being constructed as programs running on blockchain nodes and can be issued among untrusted, anonymous parties without the involvement of any third party. [11]

Smart contracts inherit properties of underlying blockchains which include an immutable record of data, and the ability to mitigate single points of failure. Smart contracts can also interact with each other via calls. Unlike traditional paper contracts that rely on middlemen and third-party intermediaries for execution, smart contracts automate contractual procedures, minimize interactions between parties, and reduce administration costs. [12]

Due to the ease of deployment, smart contracts on public blockchains or "public smart contracts" have attracted a wide variety of commercial applications. While smart contracts on permissioned blockchains or "permissioned smart contracts" are more often used in collaborative business processes since they have the potential to prevent unwanted updates, improve efficiency and save costs. One of the best things about the blockchain is that it is a decentralized system that exists between all permitted parties, and there's no need to pay intermediaries and it saves you time and conflict. With the development of blockchain technology, smart contracts are being constructed as computer programs running on blockchain nodes and can be issued among untrusted, anonymous parties without the involvement of any third party. [11]

A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties. The operation of smart contracts can hardly be decoupled from the underlying blockchain. The state of a blockchain is updated when a valid transaction is recorded on a chain, and smart contracts can be used to automatically trigger transactions under certain conditions. We categorize smart contracts to public smart contracts and permissioned smart contracts according to the blockchain platforms they operate on. [13]

continued next page . . .

Smart contracts can:

- Work as 'multi-signature' accounts, releasing funds only when a certain percentage of users agree
- Manage agreements between users
- Provide utility to other contracts (similar to how a software library works)
- Store information about an application, such as domain registration information or membership records.

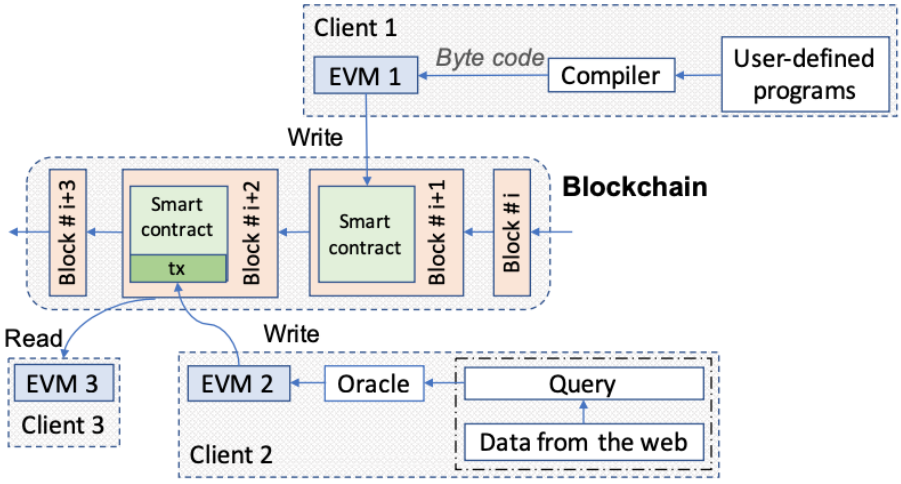


Diagram of the working of Ethereum smart contracts. [11]

In Step 1, Client 1 creates a smart contract for voting in a high-level language, e.g. Solidity. This smart contract is compiled into machine-level byte code where each byte represents an operation, and then uploaded to the blockchain in the form of a transaction by EVM 1. A miner picks it up and confirms it in Block #i+1. Once a voter has submitted his vote via the web interface, the EVM 2 queries the data from the web and embeds it into Transaction tx and deploys it to the blockchain. The state of the voting contract is updated in Block #i+2 with the confirmation of transaction tx. If Client 3, the coordinator, later wants to check the states stored in the contract, they have to synchronize up to at least Block #i+2 to see the changes caused by tx. [14]

Before permissioned smart contracts become ready for wider adoption in business procedures, many fundamental issues are yet to be solved. Notably, there is still a lack of formalized ways of composing smart contracts to suit various design purposes, especially when legal contents are involved. From a legal perspective, there is a lack of regulation and policies over smart contracts. It is sometimes hard for blockchains and smart contracts to obtain government approval. By now there is still the issue of

enforceability and jurisdiction with this technology. When evaluating opportunities, organizations should carefully evaluate the effect of such a lack of government acceptance. [11]

In conclusion, A blockchain-based smart contract or a "smart contract" for short, is a computer program intended to digitally facilitate the negotiation or contractual terms directly between users when certain conditions are met. With the advance in blockchain technology, smart contracts are being used to serve a wide range of purposes ranging from self-managed identities on public blockchains to automating business collaboration on blockchains.

4. DESCRIPTION of DIGITAL SIGNATURES

As its name implies, a digital signature is a digital representation that mimics the process of manually signing a contract with a signature in ink. [15] Given that "wet" signatures in ink can be forged, or the contract to which a "wet" signature is affixed can be altered without the signer knowing, an unalterable digital signature provides a more secure way to show that a party has indicated their agreement to adhere to the terms of a particular contract. [15] More specifically, while a "wet" signature on a paper document cannot meet the five criteria for an effective signature, a digital signature scheme can meet those criteria as listed here:

1) Authenticity;
2) Unforgeability;
3) Non-reusability;
4) Inalterability of the underlying document; and
5) The inability for the signature to be repudiated by the signer. [16]

A modern digital signature scheme uses asymmetric cryptography with a public and private key to prove that a party signed a unique contract. [15] In its most general sense, this is done by the signor encrypting a document with her private key, then anyone being able to decrypt the resulting ciphertext with a public key, thus proving the signer signed the specific document. [16]

Asymmetric cryptography is the basis of a digital signature.

In 1976 Diffie and Hellman derived the asymmetric concept of a public and private keys as a solution for securely exchanging a symmetric encryption/decryption key. [17] The Diffie-Hellman scheme involves the traditionally named Alice and Bob wanted to use symmetric cryptography in order to keep messages passed between then secret. In order to do so, they need to exchange a symmetric key that each of them will use to both encrypt and decrypt messages to and from each other. Securely exchanging this symmetric key is the aim of the Diffie-Hellman scheme. Here is how it works (each number chosen is "large" meaning perhaps in the range of 264 or greater):

1) Alice and Bob agree to a prime number n, and a primitive root modulo g;
2) In secret, Alice picks her secret key x and Bob picks his secret key y;
3) Over an unsecured channel, Alice then sends Bob (g^x) modulo n;
4) Bob sends to Alice (g^y) modulo n;
5) Using her secret key again, Alice then computes $(g^y)^x$ modulo n which is the symmetric key; and
6) Using his secret key again, Bob then computes $(g^x)^y$ modulo n which is the symmetric key.

Because $g^{xy} = g^{yx}$, Alice and Bob have generated the same, secure symmetric key by exchanging encrypted messages over an unsecure channel. Now then can encrypt and decrypt messages to each other using the derived symmetric key. [17]

Recognizing that Diffie-Hellman allows for a “man-in-the-middle” attack due to its lack of authentication, in 1978, Rivest, Shamir and Adelman (RSA) built upon the work of Diffie-Hellman to develop the basis of a modern digital signature by adding authentication by way of signature using private keys to thwart a man-in-the-middle attack. This bolsters authentication by way of having the signer of a document first hash the document, then sign/encrypt that document hash with their private key. This would allow any third party to use the signer’s public key to decrypt the message, then check the hash of the signed documents against the document itself thus verifying both that the signer is the only person who could have signed the document, and the signer cannot repudiate that she signed the document. [18]

More specifically, if Signer wants to send a signed contract to Recipient, she can produce a hash value of the contract using an agreed to hashing algorithm, then raise that hash value to the power of her private key derived as part of the RSA scheme then take the value of the result modulo the modulo value derived from the RSA scheme, and attaching the resulting "signature" to the contract (if she wants the contract itself to remain secret, she can encrypt the whole signature package using the recipient’s public key so that only the recipient can decrypt it). When the recipient receives the signed contract, they use the same hash algorithm and (after decrypting the package with their own private key if it was encrypted with their public key) raise the signature to the power of the signer’s public key derived as part of the RSA scheme then take the value of the result modulo the same modulo value derived from the RSA scheme, and comparing the resulting hash value with the contract's actual hash value. If the two agree, the recipient can be confident that the signer of the contract was the Signer, and additionally that the contract remains in its initial state. This works because as Diffie-Hellman showed, exponentiation is commutative.

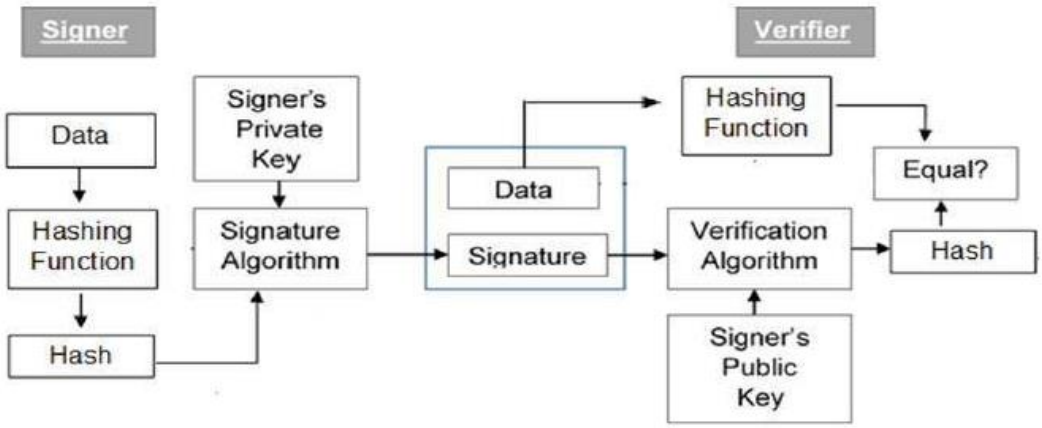


Diagram of general flow of a digital signature. [19]

How do we know who signed an Ethereum blockchain-based smart contract?

In addition to Bitcoin, the Ethereum blockchain platform uses a digital signature scheme that builds on the overall theme of RSA, but uses elliptic curve cryptography (ECC) to derive its keys as opposed to RSA's reliance on the multiplication of two prime numbers. Elliptic Curve Digital Signature Algorithm (ECDSA) provides similar service as other digital signing algorithms, but using ECC provides ECDSA with greater efficiency, particularly requiring smaller keys to provide equivalent security.[1]

Conveniently, public and private ECC/ECDSA keys are generated when a user creates a new Ethereum account. These initial keys can then be used to support smart contracts because in addition to providing access to the Ethereum ledger, the private key serves as a basis for digital signatures (while a hash of the public key serves as the Ethereum address of the ledger, and ECDSA digital signatures in Ethereum also support smart contracts outside of the blockchain. The details of the mathematics behind ECC and ECDSA are beyond the scope of this Report. Suffice it to say, that as modulus obscures the calculation of of keys in RSA, so does mathematics associated with the algebraic structure of elliptic curves over finite fields. [20]

In order to digitally sign a smart contract on top of the Ethereum platform, the signer first creates a hash of the contract using the Secure Hash Algorithm SHA256 which is the standard hashing algorithm used by Ethereum. This results in a 256-bit value representing the contracts. Then Ethereum's JSON Remote Procedure Call (RPC) is used to tell the signer's underlying Ethereum wallet to used the signer's private key to sign the smart contract. [20]

In conclusion, using the well-proven power of ECC to generate secure private keys, and similarly sturdy ECDSA to use those keys for a secure digital signature, smart contracts on top of the Ethereum platform can be relied upon today as properly signed by the signing party as the digital signature protocol is well-designed and has survived attacks from academia and malicious actors. However, Ethereum's digital signature protocol does rely on the secured privacy of the signer's private key, thus any breach of the security of these keys would allow malicious actors to undertake seemingly unassailable signatures on behalf of another party. Further, in addition to persistent rumors that the National Security Administration has inserted a backdoor of sorts in the ECC protocol, quantum computing and its potential to unlock new avenues of brute-force and other calculations could eventually provide a crack in the security provided by ECC/ECDSA. Although cryptography has made great strides since RSA was developed in the 1970s, history has proven that no cryptographic solution is secure *forever*.

[1] A 256-bit elliptical curve key is as secure as a 3072-bit RSA public key. [25]

5. DESCRIPTION of FORMAL VERIFICATION PROTOCOLS

This section discusses a novel formal modeling approach to verify a smart contract behavior in its execution environment. This technique is applied to a concrete smart contract example and analysis of its breaches with a statistical model checking approach has been completed. [21]

An example of a simple decentralized application implemented as an Ethereum smart contract in Solidity. [21] This contract behavior along with its users and the execution environment are then formally modeled. Based on the simulated executions of these models, the vulnerability of the smart contract is displayed and alternative designs are proposed.

Example of a name registry smart contract on Ethereum is used. [21] The goal of this contract consists in associating a blockchain account address to a unique username. This kind of service could be used for instance to simplify currency transfer between accounts by using simple and custom usernames instead of a long and complicated address. Another way to use it would be to declare and manage web domain names. The Solidity source code of such a contract is given below [21]:

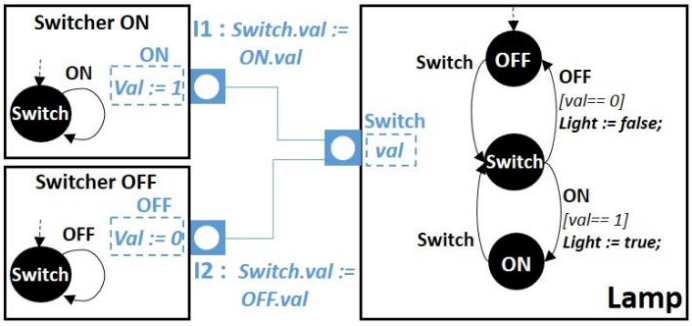
```
pragma solidity ^0.4.11;
contract NameReg {
    mapping (string => address) private
        nameToAddress;
    event Notify(bool _success);

    function register(string _name) public {
        if (nameToAddress[_name] != address(0)) {
            _success = false;
        } else {
            nameToAddress[_name] = tx.origin;
            _success = true;
        }
        Notify(_success);
    }
}
```

At first glance, this simple code could seem robust: once a user has proposed registering a name using his account, there is no chance that his name is stolen. However, this is not true: by modeling this smart contract, users along with blockchain behaviors and simulating their executions, vulnerability of this implementation can be displayed.

Modeling and Verification Environment:

BIP (Behavior Interaction Priorities) framework is used for its strong component-based modeling formalism and its statistical model checking engine for systems verification. [21]

Atomic Components: They are used to model systems elementary behaviors. They are finite-state automata, extended with variables and ports. Formally an atomic component is defined by: A finite set of states; A set of ports; A set of variables, partitioned into two sets T and U for timed and not timed variables; A set of transitions.



Component based modeling with BIP. [21]

Components Interactions: Composite components are defined by assembling sub-components (atomic or composite) using connectors. Connectors relate ports from different sub-components. They represent sets of interaction patterns that are non-empty sets of ports that have to be jointly executed. For every interaction, the connector provides the guard and the data transfer to exchange data across ports involved in the interaction.

The Statistical Model Checking Tool (SMC): The BIP framework is currently equipped with a series of runtime verification and simulation engines. [21] The SMC tool is a model checking procedure to decide whether a given system B satisfies a property φ, in order to estimate the safeness of the system. Properties are logical operators described using the PB-LTL (Probabilistic Bounded Linear Time Logic) formalism. Statistical model checking refers to a series of simulation-based techniques that can be used to answer two questions: First question is Qualitative: Is the probability for B to satisfy φ greater or equal to a certain threshold ϑ? Second question is Quantitative: What is the probability for B to satisfy φ? Note p the probability we want to estimate, SMC determines an estimation p', with a precision δ and a risk level α, so as: $\mathbb{P}(|p' - p| \leq \delta) \geq 1 - \alpha$. SMC uses SPRT (Sequential Probability Ratio Test) and SSP (Single Sampling Plan) algorithms to determine the number of simulations to reach a verdict.

continued next page . . .

Smart Contracts Specification:

Generic smart contract specification: A smart contract can be modeled as an atomic or a compound component upon its complexity.

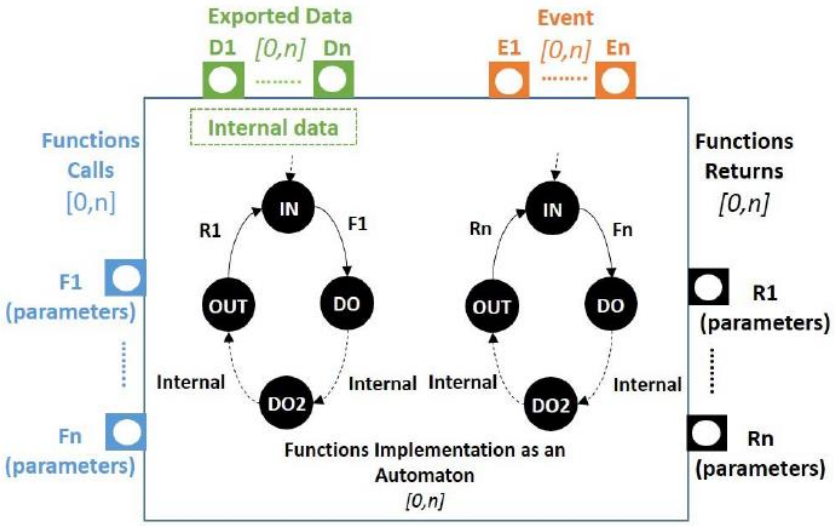


Diagram of generic smart contract specification. [21]

The Register smart contract: The register contract component is defined by its Interface and Behavior.

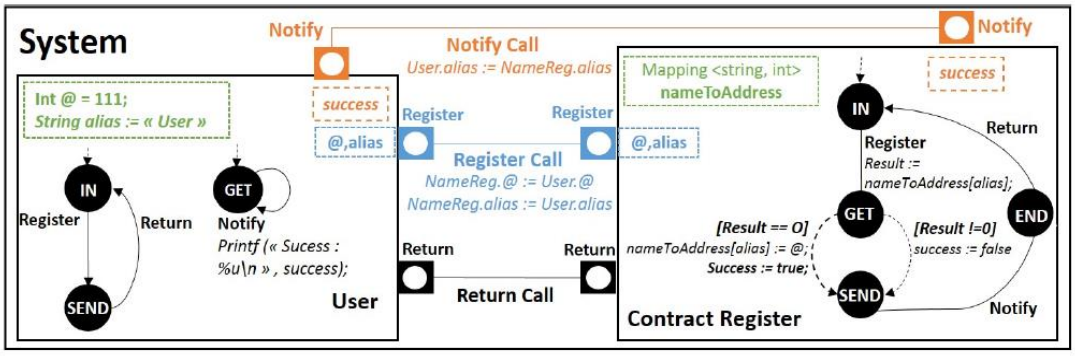


Diagram of user executing the register smart contract. [21]

Verification of the Smart Contract Behavior: In order to build robust smart contracts and verify their safety against external attacks, it is essential to take into account the blockchain's behavior properties.

Modeling the Blockchain Behavior: The blockchain compound component enables external interactions through ports inherited from its internal components. Port $GET\ Tx$ receives transactions from users and data ports $Pending\ Tx$ and $Blocks$ exports the pending transactions list and data blocks.

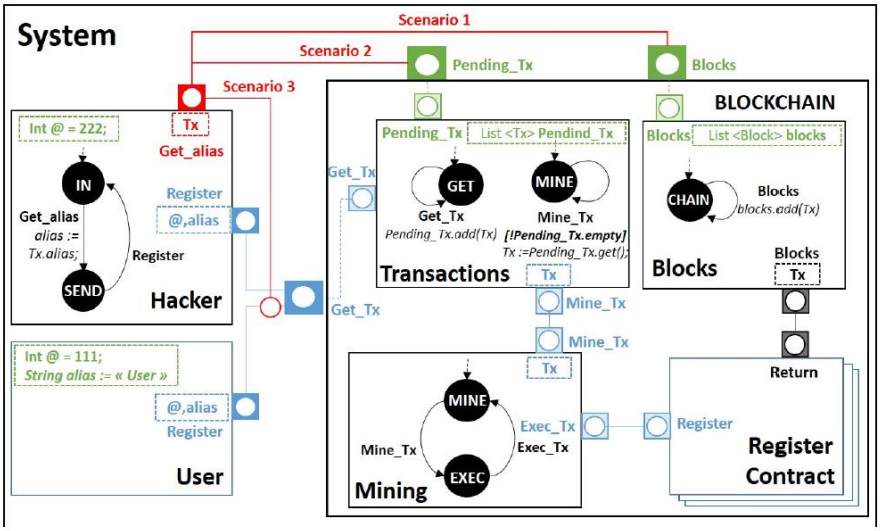


Diagram of Smart Contract execution with blockchain and hacker behaviors. [21]

Component Transactions models the reception of external transactions. It stores them in the pending transactions list until they are retrieved randomly for mining. Component Mining models the overall mining process. It mines pending transactions and sends execution commands to corresponding smart contract. For simplification purpose, we don't take into account the number of miners and the type of consensus. Component Block encapsulates contracts execution results into blocks and exports them through the port $Blocks$. Contracts components executes the register function calls and sends the execution result to the Blocks component.

Verification Results: Analysis of the safety of the register smart contract execution by introducing a hacker behavior model is done. The hacker's purpose is to steal the user's identity by registering their alias with his own address. Three scenarios are possible regarding blockchain interface.

Scenario 1: He retrieves the name after the registration of the user from the mined blocks.

Scenario 2: He retrieves the name from the pending transactions data when the user transaction is not mined yet.

Scenario 3: He gets the name from the network, that is, directly from the user call interaction with the blockchain. Statistical Model Checking (SMC) probability evaluation feature with parameters $\alpha = 0.1$ and $\delta = 0.1$ is used [21].

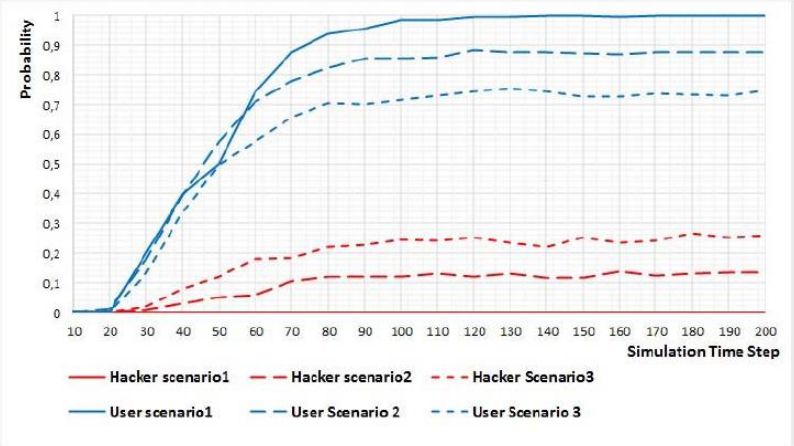


Chart showing Statistical Model Checking Results. [21]

In scenario 1, the hacker can never succeed in registering the user name. The register contract rejects the registration because the alias is already registered in the AddressToAlias list. In scenario 2, the hacker has an average of 12% to hack the register. In scenario 3, the hacker has an average of 25% to hack the register. Analysis of the execution traces showing the success path of the simulations is also done. In scenario1, the register smart contract behaves correctly and rejects the hacker tentative to register the user's name, which is already registered in the contract AddressToAlias map. In scenario2, the hacker succeeds when both the hacker and the user transactions are in the pending transaction list. Due to the random mining, the hacker transaction can be mined and executed first. In scenario3, the hacker intercepts the transaction while the user sends the register call transaction. Since in scenario2, the hacker should wait for the user's transaction to be in the pending transaction list, it explains the smaller chance for the hacker to succeed than in scenario3. In order to avoid such attacks, the registration process could require two steps. First the user registers the hash of the name and then only he registers the actual corresponding name in a second transaction. [21]

In conclusion, a novel modeling formalism with strong semantics for smart contracts and blockchain properties has been used. By applying this formalism on a concrete example of smart contracts, modeled its behavior and interactions with its execution environment also represented using the same approach. The simulation of these behaviors in the BIP framework and the analysis of its results using SMC allowed to reveal scenarios where the smart contract behavior can be breached by hackers.

6. APPLICATIONS of SMART CONTRACTS in the REAL WORLD

This section gives an insight of how the smart contracts are being used in the real world and also discusses in detail of how smart contracts can be applied in various areas making people's life easier. In addition the areas where blockchain can be implemented which helps to get things done in an efficient way is discussed.

Smart Contracts in Ethereum:

Ethereum is a new technology made possible by public blockchains, smart contracts are difficult to understand because the term partly confuses the core interaction described. While a standard contract outlines the terms of a relationship (usually one enforceable by law), a smart contract enforces a relationship with cryptographic code. [22]

Put in a different way, smart contracts are programs are executed exactly in the way they are written by their creators. For example, an ethereum user can send ten ether to someone on any date using a smart contract. In such cases, the user needs to create a contract, and have to push the data to that contract so that the desired command could be executed. Ethereum platform is built for creating smart contracts specifically. However these new tools aren't meant to be used in isolation. The building blocks for 'decentralized applications' can be formed using smart contracts and also the whole decentralized autonomous companies.

How smart contracts work in Ethereum:

Bitcoin was the first to support basic smart contracts in the sense that the network can transfer value from one person to another. [22] The transactions are validated by the network of nodes only if certain conditions are met. However, bitcoin is limited to the currency use case.

In contrast, Ethereum replaces Bitcoin's more restrictive language and replaces it with a language that allows developers to write their own programs. Ethereum allows developers to program their own smart contracts, or "autonomous agents," as the Ethereum white paper calls them. The language is 'Turing-complete', meaning it supports a broader set of computational instructions. [22]

Smart contracts can:

- Function as 'multi-signature' accounts, so that funds are spent only when a required percentage of people agree.
- Manage agreements between users, say, if one buys insurance from the other

- Provide utility to other contracts (similar to how a software library works)
- Store information about an application, such as domain registration information or membership records. [22]

Benefits of Smart Contracts:

Smart contracts make different processes run automatically. You do not need to supervise the fulfillment of the contract; mathematics does it for you.

Hence, owing to smart contracts, processes become:

- Automatic – Whatever is defined in a contract will happen
- Fast – It will happen within seconds
- Direct - As no intermediaries are involved
- Cheap – No fees are to be paid
- Transparent – All the information will be registered on the blockchains. [22]

10 Areas of Smart Contract Application:

1) **Digital Identity:**

Today, different details about your life are held by different institutions – demographic facts, ownership rights, job details,bank records etc. To maintain all this information in one file, we would need to carry a huge set of documents. This turns out to be pretty inconvenient, especially if you want to pass an identity verification.

This problem is solved by smart contracts and it also allows to maintain all the data of a single person in one place. In case something happens to you, it would be immediately registered on the blockchain for maintaining your identity holistic. Owing to this,instantaneous verification of KYC is possible. The privacy would never be affected as you are the one to decide what information can be disclosed.

2) **Banking:**

The customers always feel annoyed whenever they are trying to transfer money, as they have to cover a fee, pay a percentage of the amount they send, and then they need to wait for a few days until the transaction is processed? The imperfections of the modern banking system are hard to deny even though their work rather smoothly. Smart contracts do not require any intermediaries. Therefore, customers need not pay any fees. Transactions become fast and cheap as there's no bureaucracy involved. Moreover, the transparency guaranteed by the blockchain reduces the possible risks of fraud.

3) Tax Records:

An image of a guy running to a tax office with a pile of paper just to find that it's closed is a source of endless jokes in cartoons. But when you are in the shoes of this guy, you're unlikely to laugh.

Smart contracts would save you from fines as payments are triggered automatically and prevent you from committing an unintentional crime. Also at the same time, complete data about taxes is being recorded on the blockchain and is available for everyone who is determined enough to check the database. This transparency of the tax records makes cheating impossible.

4) Insurance:

When we get into a minor car accident, the first thing we would worry about is an insurance payment. If the accident is not our fault, we expect the guilty side to cover repair expenses. But what would we do if this person denies their fault? Our chances of getting a refund are not too inspiring.

May the car be equipped with an IoT device reporting its location, speed, time of the accident, we would have no reasons to worry. In case we're right, the data on the blockchain would prove our words, and we would get our payment automatically.

5) Real Estate and Land Titles Recording

Real estate deals are painful for a normal person to handle, especially cross-border ones. You don't want to get stuck or involved in the time consuming legal negotiations which can go for months and paper signings along with other bureaucratic nuances that are related to transfers of ownership rights.

But with the help of smart contracts, this pain is easily avoided. The centralized registry of the property would allow you to buy and sell real estate without intermediaries and to pass ownership rights within minutes. You could find the apartment you like in a few clicks, pay for it, and at the same time get the proof you're the new owner. Meeting with the seller isn't required .

6) Supply Chain

Whenever you visit a store to buy seafood, you can't know if it 100% fresh. The store may display that the items have just arrived. However, it's tough to believe. IoT devices when combined with smart contracts are going to make a revolution in logistics and supply chain. Tracking of the way products pass before they reach the retail spot will be automatic and transparent with the help of smart contracts. The location of the goods are known at any moment of time along with the conditions they're stored in and also provides the time they would arrive at. For example, this technology can also be used for tracking any type of goods such as retail goods, oil, responsibly sourced coal, gold, etc. Due to the use of blockchain the vendors are more likely to become trustworthy and the risks of fraud decline.

7) IoT(Internet of Things):

When you return your home from work, and a box with the products you need is in front of your door. When you switch on the TV and you see the movie you are interested to watch is already downloaded. Your curtains open automatically a second before the alarm is about to ring. Due to smart contracts a smart house is no longer fiction, and as a result it becomes more automatic and reliable. IoT is one of the most inspiring smart contracts examples as it is tightly connected with our daily routines.

8) Authorship and Intellectual Property Rights:

Entertainment industry is continuously plagued with the issues of piracy and authorship violations. Writers, photographers, musicians and other artists are deprived of their royalties due to the fraudulent exploitation of their intellectual work.

Smart contracts can improve this situation by maintaining a transparent registry of authorship on a blockchain and in turn providing an efficient solution. For example, when your novel or stock photo is downloaded by someone, you get a refund automatically. In addition, the copyrights are registered securely, and they can't be alienated by anyone.

9) Life Science and Health Care:

If you wear a health tracking device that keeps track of your blood pressure and heartbeat which also transfers this collected data to a blockchain regularly. In case if any of the indexes exceeds the norm, you receive a notification on your phone triggered by a smart contract. This way, you will be warned when something goes wrong. This gives you enough time to take the medicine and helps you prevent a crisis. Moreover, a blockchain guarantees the privacy which makes it convenient for secure storage of clinical trials results.

10) Gaming and Gambling:

The Internet is filled with multiple offers for playing online games or for checking out new virtual gambling platforms. You usually don't care whether you win or lose when you play it for free. However you start thinking about payments when you play the game for money and also think of ways for getting your winnings.

Now, if a virtual gambling platform adopts smart contracts, then there would be no reason to worry: you get your reward whenever you win and you can't fool the system when you lose. Gambling becomes transparent and honest. This algorithm can also be applied to any paid computer game and e-sports.

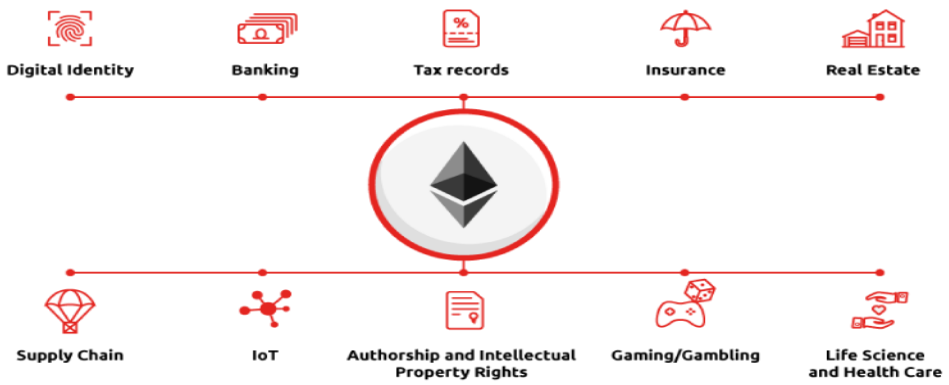


Diagram of the various areas that can utilize Smart Contracts. [23]

Further Applications of Blockchain:

Blockchain has been in the spotlight after successful boom of the Bitcoin. There have been efforts to leverage salient features of Blockchain for different applications and use cases. This section presents a comprehensive survey of applications and use cases of Blockchain technology.

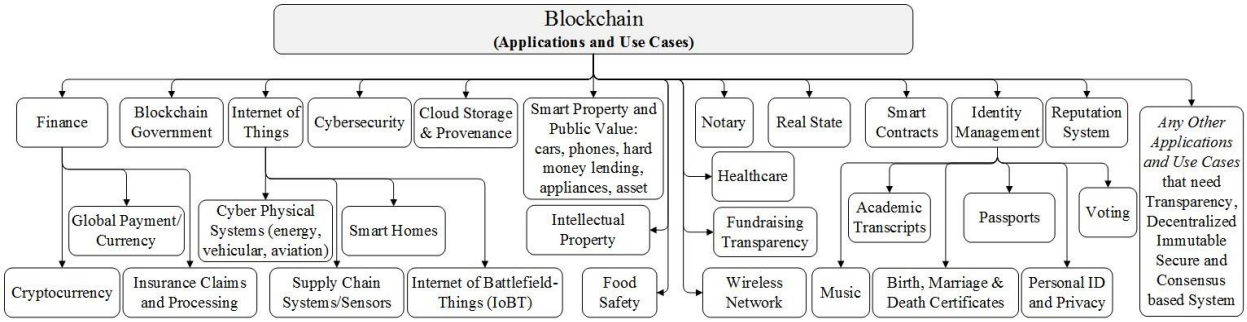


Diagram of the use cases and applications of Blockchain.[24]

Finance:

Conventionally, the financial transactions are verified and processed by intermediaries such as banks.. Having such a centralized system puts immense work in the hands of intermediaries, meanwhile the transactions are prone to errors as multiple uncoordinated parties are required to keep the record and adjust them. [24] Therefore, this makes the entire process costly and time-consuming. The Blockchain simplifies such complications associated with financial services by introducing a distributed public ledger, where the transactions are verified by the miners using “proof-of-work.” [24] Since, each node in the Blockchain network has a copy of the updated Blockchain, there is transparency regarding the transactions. [24] Since,

the blocks are chronologically arranged, once a block is added to the Blockchain with a verified transaction, the entire Blockchain is immutable. [24] Thus, a transaction cant be manipulated once it is registered into the system. In case of a conflicting Blockchain where branching might occur, miners always go for the longest chain, as the longest chain is more reliable. [24] Having this type of secure robust verification method and communication protocol, it creates an effective system to improve our existing financial services.

- Cryptocurrency
- Global Payments
- Insurance claims and Processing

Blockchain Government:

Government organizations and units can use Blockchain technology in order to build the most trustworthy and effective government operations using collaborative and transparent networks. Blockchain technology known for its salient features will help achieve accountability, transparency and trust among stakeholders such as government officials, leaders, citizens and their different operations. In In order to address the accountability of its bodys, government is required to make its affairs transparent. To achieve this, the government possibly have to make a great amount of data accessible and open to the public. As per reports, making data accessible to the public in the Internet could benefit the people in an order of trillions of dollars. The illegitimate doings can be exposed by several entities using open data. The public can question the quality of health-care, and food supplies with given open data which eventually makes the system fairer and trustworthy. [24] Hence, making the data available to the public is helpful for the economy, but it also has its own challenges. However, the data is left unnoticed by the public if it is released only once a year. Thus, an alternative to this can be a Blockchain government, where the data is distributed in the public ledger, and is open to the public all the time. [24] Moreover, smart contracts can be used to ensure the electorates work in favor of the electors. [24] The contracts can be based on the manifesto of the electorates, and they only get paid once they meet the demand of the electors via the smart contracts. [24] This kind of technology allows to keep electorates in check and also enforces them to fulfill their promises.

Blockchain Health-care:

Sensitive information like personal health records needs to be dealt with high security. Such personal records can be encoded and stored using Blockchain and provide a private key which would allow only specific individuals to access the records. [24] Similarly, the same protocol can be applied to conduct research where personal records are used under the HEALTH INSURANCE PORTABILITY AND

ACCOUNTABILITY ACT of 1996 to ensure confidentiality of the data. [24] Patients records in the Blockchain can be automatically sent to the insurance providers or doctor can send the health record to concerned parties securely. [24]

Food Safety:

Food safety is one of the most critical issues to be addressed since over 0.6 billion (equivalently 1 in 10 people) in the world become ill after consuming bad food every year. [24] Around 1,167 people die due to this bad food every day . To prevent these issues, Blockchain technology can help to prevent counterfeiting issues for food to have visibility across the food supply chain and help to access any information such as food content, its origins, expiration, etc. in seconds. [24] Food consumers will better control over food and information with high accuracy and transparency for food safety. [24]

Internet of Things (IoT):

The electronic devices that are getting connected to the Internet are increasing rapidly in number every year. As a result this gives way to the Internet-of-Things (IoT) due to the massive number of devices interlinked to each other. The IoT is expected to transform the way of lives where ideas like smart homes is feasible [24] While this new phenomenon is likely to make lives easier, having massive number of heterogeneous devices connected to the Internet creates graves issues regarding cyber security and privacy. [24] The Blockchain can be an important technology to secure IoT because Having millions of devices connected to each other and communicating, it is important to ensure that the information flowing through IoT remains secured and makes the participants accountable. [24]

- Energy Cyber Physical System
- Vehicular Cyber Physical System
- Supply Chain Systems/Sensors
- Smart Homes
- Blockchain in Aviation Systems

In conclusion, today, numerous computer scientists work on projects aimed at solving the acutest problems with the help of the blockchain technology and smart contracts. After all, no one said it would be easy to change the world and to enhance it with new, better ways of solving common problems. There is a lot of work ahead for blockchain engineers and smart contract developers. Eventually smart contracts will

be incorporated into various fields making the work carried in those fields in a trustful, transparent, secure and in a confidential way by maintaining integrity

Further, smart contracts use cases are not limited to the ones discussed in this paper. Anyone can come up with any idea of how smart contracts on a blockchain can be exploited. For instance, keeping track of academic performance at schools. And it's just one way of vast number of possible implementations.

7. ISSUES FACING DEVELOPERS WRITING SMART CONTRACTS

This section will talk about the different kinds of problems and issues that a developer should be aware of while developing a contract for Ethereum. These problems can be due to the inherent structure of the blockchain itself or due to the coding practices followed by the developer. Some of the main issues with the structure of blockchain are mentioned below.

1. The Majority Attack

The Proof of Work [25] phase, also known as 'mining', depends heavily on the computing power and the amount of work done by a miner. Because of this loophole, sometimes miners can work together in groups to mine more and more blocks in the chain. [25] This will cause them to take over the control of the network and modify any block in the chain according to them. By controlling the majority of the network, they can also deny other miners in the network and have complete control over the chain.

2. Forking inconsistencies

Sometimes there can be updates to the consensus rules in the network but only some nodes would be updated and others not. This will cause an inconsistency in the way these two sets of nodes process the blocks on the chain. There are mainly two types of forking issues that occur - Hard fork and Soft fork. [25]

3. The scale of the blockchain

As the chain becomes bigger and more complex, the time and resources needed to very and maintain the blocks in the chain increase drastically. This creates a lot of issues with the performance of the network over such a big blockchain.

4. Current regulations and rules

Since blockchain is a decentralized network to do transactions, removing the middlemen from the transactions, it does not sit well with the government and the centralized banks. It creates problems of tracking money and transaction for these entities and reduces the bank's ability to control the economic policy. [25]

5. Cost of infrastructure

Every miner in the blockchain network will be fighting to finish the PoW and PoS [25] tasks in the blocks so that they can make monetary gains from them. To be able to achieve this, they should be the

fastest in doing these tasks. So, they would have to invest heavily in heavy and complex infrastructure to make their computations faster.

Let us go through some of the coding issues and challenges for developers on the Ethereum platform:

1. **Reentrancy on a single function**

This is the most famous and common types of vulnerabilities the attackers try to exploit in a blockchain network. This happens when a second function call is made even before the first invocation of the function finished its execution. As seen in the example below, if the attacker can call withdraw function multiple times, and each invocation is started before the statement to set the balance is executed, they can keep withdrawing money. This exact vulnerability was exploited during the DAO attack in 2016 where attackers stole $50 million worth of ether. [26]

```
// INSECURE
mapping (address => uint) private userBalances;

function withdrawBalance() public {
    uint amountToWithdraw = userBalances[msg.sender];
    (bool success, ) = msg.sender.call.value(amountToWithdraw)(""); // At this point, the caller's code is executed,
    // and can call withdrawBalance again
    require(success);
    userBalances[msg.sender] = 0;
}
```

Diagram of double reentrancy function. [26]

2. **Front running**

Here the attackers can use their knowledge about future actions yet to be executed in the chain and using them, modify the block itself. For example, if the attacker knows that there is an action to perform the delivery of something at the end of the contract, they can call that directly by modifying the structure of the block. By doing this, they would bypass the payment part. [26]

3. **Integer Overflow and Underflow**

In computer systems, there is a maximum value for the integer types. For example, uint has a max value of 2256. If we go over this, it will reset to 0. So the developers should take care of these situations where the range of the data type could be used to manipulate the data held by that variable. [26]

4. **Using revert functions**

This is a special case in contracts which have a refund mechanism built into them. If programmed without any proper checks to the revert or refund functions, this can lead to the attacker to keep using the revert function to get more money than they had given. [26]

5. **Insufficient gas griefing**

Attackers can pass another void contract into a contract in the chain. When the main contract calls the void contract and it fails, the main contract could also fail. This can be taken care of by proper handling of errors in any sub-contracts of the main contract on the chain.

A major issue faced by the developers of Blockchain is the performance issue related with it. The section above talked about challenges with scaling of the Blockchain and also about the cost of the infrastructure to run and participate in the chain. Developers of the Blockchain have explored many ways to reduce the computational cost to make this technology more cost effective and scale better. There have been several developments in the Ethereum framework to tackle these issues, which are discussed shortly. But researchers believe there is still a lot of scope for improvement in the consensus model to improve performance. One of the approaches provided by Naoki Shibata [27] talks about how we can reduce the work done during consensus by introducing a new type of model called "Proof-of-Search" (PoS).

The "Proof-of-Search" Consensus Model

This is a consensus model developed by Naoki Shibata based on the Proof-of-Work consensus model already available in the Blockchain ecosystem. The main aim with this consensus model was to reduce the computational needs of Proof-of-Work model by using the wasted computing power to search for a solution to an optimization problem. [27] This solution can just be an approximate solution which can be achieved with the computing power left over from that stage.

Typically, in a general model of the Blockchain, we have miners and validators. The miners will follow one of the consensus models and validators will validate the work done by miners and add the new block to the chain. In the PoS model, there are 4 new entities - Evaluator, Searcher, Client and Job. An Evaluator is a computer program that deterministically the evaluation value of a solution for a given optimization problem. [27] The Evaluator is platform agnostic. A Job is a data that represents a request to search for a solution to an optimization problem. Any node that submits a job is a Client. A Searcher is randomized search algorithm that searches for a solution.

The figure below shows the structure of a minimal PoS scheme. [27] The way the PoS scheme works is that each block proposed to be added to the network by a miner has to include a hash and nonce

which is made up of a very good solution to a problem already present in the network. The miner has to create a new block's hash value that begins with a required number of zero bits which is similar to Proof-of-Work. But where PoS differs is in the creation of the nonce value. To generate a nonce value, a miner has to execute many solution candidates with an evaluator. A valid nonce in this scheme is made up of a solution candidate and the value of that candidate from an evaluation. [27] By providing a nonce value that makes a block's hash value begin with a required number of zero bits, the system provides a probabilistic proof [27] that miners have done evaluations of many solution candidates. In this scheme, more evaluations a miner performs, better their chances of getting a good solution. The system uses the searcher to find the node with the best solution.

The evaluation scheme for a new block is as follows:

1. First check if the nonce value is formed by the evaluation of the solution candidate used.
2. Check if the hash value contains certain number of leading zeros
3. Verify the hash value

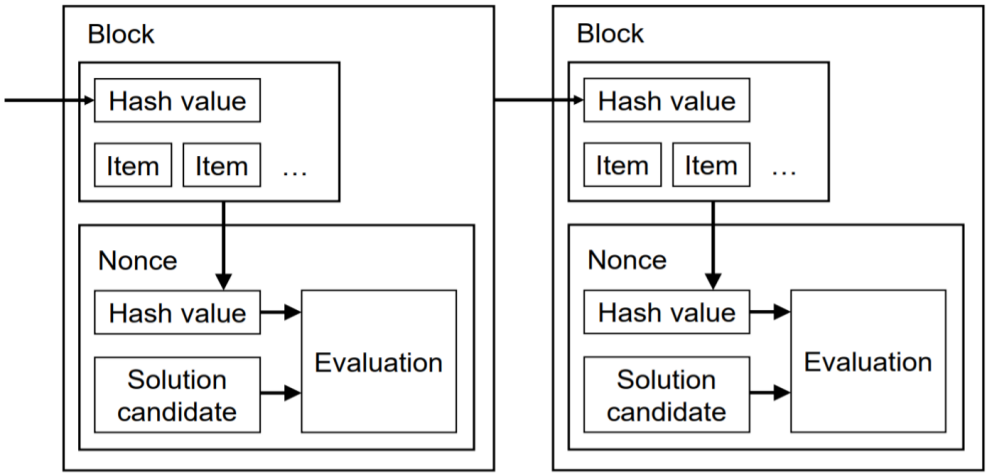


Diagram showing a minimal PoS Scheme. [27]

The problem with this minimal scheme is that is works with a single evaluator as shown in the figure. This does not have the functionality to search for the best solution. [27] To overcome this limitation and to enable the system be more computationally efficient, we can perform simultaneous execution of multiple jobs. The 3 main properties of the PoS scheme are:

1. A miner should be incentivized to find a good solution
2. A node should not be incentivized to submit a problem for which the system already has a good solution
3. Submitting of a problem that is not worth solving is discouraged. [27]

PoS scheme is actually costlier than Proof-of-Work scheme if the nodes executes one task at a time. To make it more reasonable for the miners, we can allow them to execute multiple tasks. In order to make the probability of finding the best solution proportional to the computational power spent by a node, we use mini blocks between blocks. [27] Each mini block is a job. Mini blocks are shown in the figure below. Each time a valid nonce is generated for a job from a mini block, it is broadcast to the network. After verification, that mini block is added. When all the mini blocks for a block have been added to the chain, we add the entire block to the chain. This is the basic idea of the PoS scheme to allow multiple jobs to execute simultaneously.

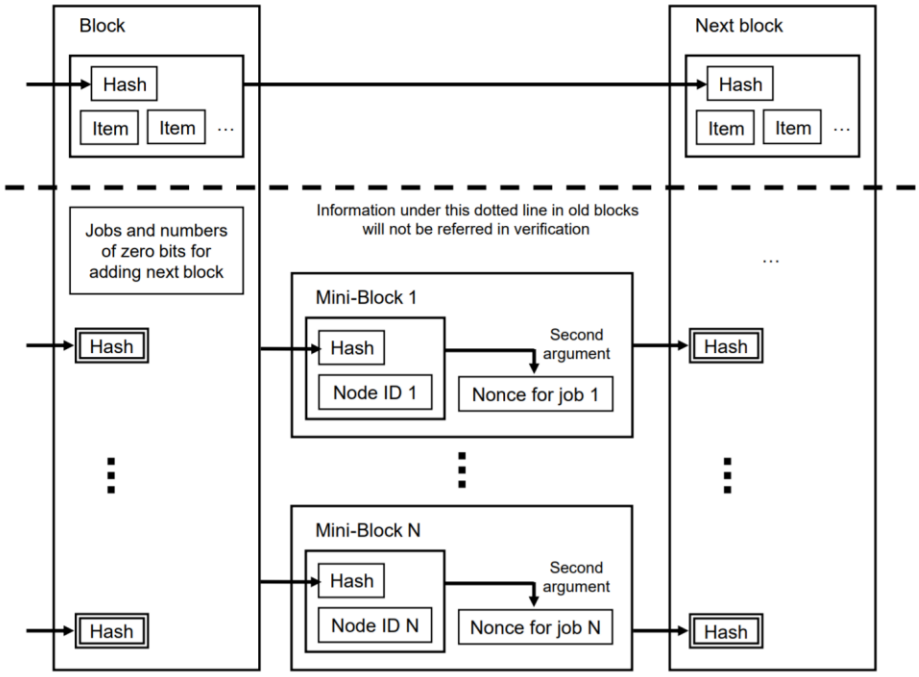


Diagram showing a PoS scheme with simultaneous executions. [27]

Some of the developments in Ethereum to tackle the scaling issues are:

1. Raiden Network: It is a network built on top of the ethereum ecosystem which allows the actual transfer sequence, where the value is transferred from one entity to another, to happen off the main blockchain network. This avoids the consensus bottlenecks in the peer-to-peer payment networks.
2. Plasma: It is a framework that allows hierarchical setup of blockchains for Smart Contracts. We can have several contracts running in this blockchain in a hierarchical setup where we can have some contracts as children of other smart contracts. So, the entire network will contain a root blockchain and then several blockchains for parents and children. The idea to use this setup is that it will be much easier to manage dependent smart contracts. With this setup, the children can periodically update the parent to maintain data integrity and concurrency.

3. Sharding: This technique divides the network into shards and the validators in the network only need to worry about validating a subset of the transactions belonging to their shard. Here, shard is just a portion of the network, which includes a certain set of validators and certain transactions for the shard. By dividing the work among different shards, developers have been able to get higher throughput from the blockchain network.

To conclude, the developer platform for Blockchains is always evolving. This technology is still at a nascent stage. The developer community is trying to perfect it to its promised state of secure transactions. But the correct framework on Ethereum and other similar frameworks, though secure enough to be used in the real world with real transactions, they do not have a well defined and well documented standards to write code. This has led to many vulnerabilities in poorly written code leading to attacks like Reentrancy attacks, Front running, Integer Overflow and Underflow. [26] To make developers more confident in writing code for Blockchain, there is a need for standardization and documentation of the frameworks on which they run their programs. Another major setback for current Blockchains is the computational need to participate in the network.

There have been developments in the form of introducing Sharding, a new concept of the consensus model in the form of PoS [27] and more. To make Blockchain to be widely accepted, the main problems to be addressed are the robustness of the code, scalability issues and computational needs of the network. Till recently participating in a blockchain meant that a miner had to build complex computational systems with huge power supply demands, many GPUs, a server grade processor and network adaptor to handle the computational load to be part of a network and also be profitable in mining. The main aim for the Blockchain community is to make mining accessible to the large masses so that many people can participate in the mining process. This will help in the distribution of load over a bigger network easier because of huge availability of processing devices. Also, this would mean that the network would comprise of relativity less powerful machines (like a laptop or home desktop), which are already in abundance, connected to the biggest network in the world - WWW. This would encourage more miners and ultimately help in reaching the goal of a blockchain network which is having a completely decentralized system to handle transactions, by cutting down the middle men and creating complete transparency in all the transactions in the network.

8. Making Smart Contracts Secure:

Vulnerabilities in smart contracts present serious issues, and require fixing for many reasons: First, application of smart contracts deployed in real life handle financial assets of significant value. While we argue about smart contract importance, it's total evaluation is 108,311,732.28 Ether (over three billion according to the Internet). Second, bugs in smart contracts (once developed) cannot be fixed (even by its creator), ie. smart contracts are immutable in nature. This is a significant step back in trying to fix vulnerabilities in smart contracts.

A significant incident in 2016, "DOA attack" exploited these vulnerabilities in smart contracts and stole a total of 3.6 million Ethers (worth $50 million at the time of attack). Another late incident by wallet rescue group failed to respond when 153,037 Ethers (worth over 30 million dollars) were stolen from large Ethereum multisig wallet contracts2 caused by Parity Multisig Bug [2] compromising three transactions in total. All these vulnerabilities were designed flaws in smart contracts and caused a great deal of loss and question trust of smart contracts and Ethereum. In this section, we will talk about how can we make smart contracts to be secure, less-vulnerable, and difficult for attackers to breach in.

FSolidM framework:

Most of the vulnerabilities attack happened in past are mostly because of semantic gaps in assumption of smart contracts. For example, reentrancy attack [28] where attackers invoke functions repeatedly before first function call is completed, in order to take control of the flow of the chain. FSolidM [14] is a graphical tool for rigorous designing of finite state architecture in Smart contracts.

Why FSolidM is good?

- Compatible with Ethereum framework and hence with smart contracts.
- Provides tool for transforming from finite state machines (FSM) to solidify language used in smart contracts.
- Also provides with plugins that implement security features and designs, that developers can use in their models.

continued next page . . .

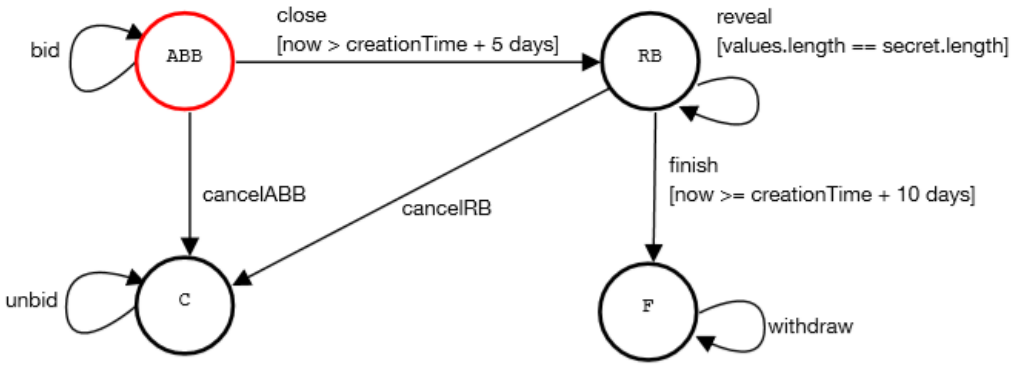


Diagram of example FSM for blinded auctions. [14]

States: AcceptingBlindedBids (ABB) - initial state, RevealingBids (RB), Finished (F), Cancellation.

Transactions: close, reveal, withdraw, unbid, finish.

All transactions are followed by semantic checks before execution. For example, during execution of reveal transaction, following check is performed (values.length == secret.length), i.e., bid is only revealed when the bid amount of secret bid in the machine matches with the user. These checks makes it hard for attackers to exploit such contracts and hence, making contracts more secure.

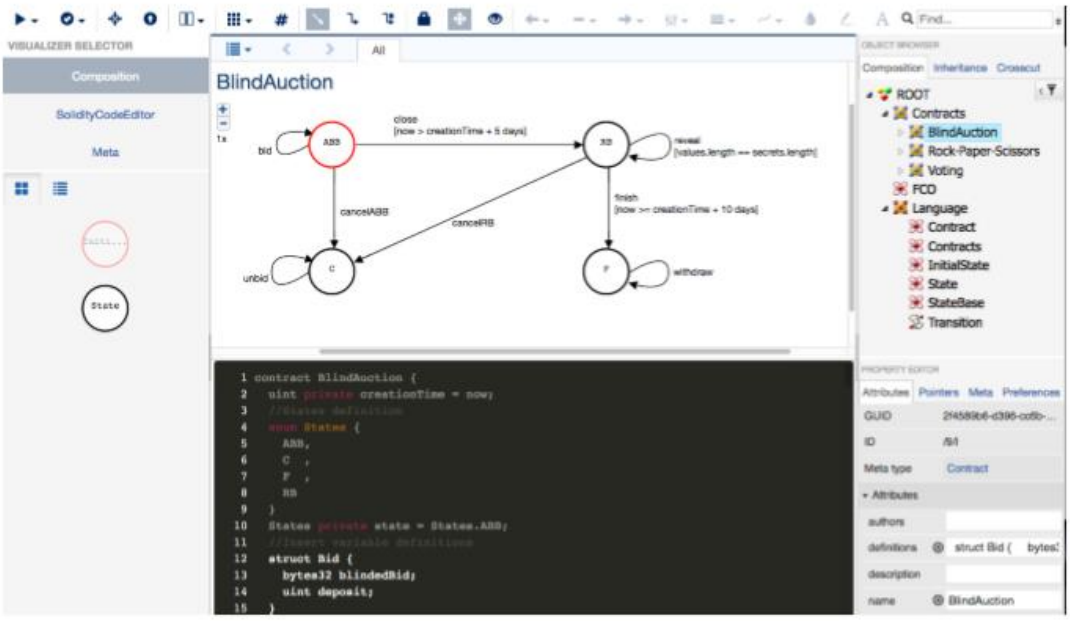


Diagram of Graphical and code editor provided by FSolidM framework. [14]

Drawback - While this framework is a secure way of designing smart contracts and should definitely be used by developers/designers in current and future smart contract designing and development, complicated system with a large number of states can be challenging to design in terms of cost and time.

A specification language for smart contracts (SPESC):

While designing and implementation restricted to developers and designers limit the intelligence in designing smart contracts which later cause vulnerability issues. SPECS is a tool that challenges this restriction and unveils new levels of designing involving non-IT smart people from the industry in designing perfect smart contracts. SPESC framework uses natural language in writing smart contracts.

The results in reference [29] demonstrates that SPESC can be easily learned and understood by both IT and non-IT users has greater potential in facilitating collaborate development in smart contracts. Also, it enables users to write smart-contracts just like writing contracts in real life makes it easy to write secure contracts by lawyers and even government without need of developers who understand software coding.

SPECSC consists of-

- Description of parties (the role each individual plays in the contract)
- Rights and obligations of each party (set of terms, and conditions each term should hold)
- For each term how the balances of accounts (e.g., assets) should change when the term holds.

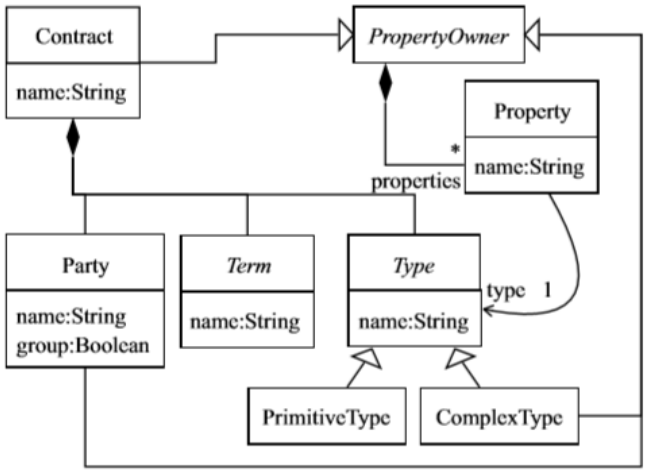


Diagram of meta mode of general structure of SPESC specialization [29]

Contacts in SPESC can have it's own contract properties (denotes a static field denoted a specific data type, refer above. It is divided in two subclasses- PrimitiveType and complexType. [29]

term *name* **:** *party* (**must**|**may**) *action*
(**when** *preCondition*)?
(**while** *transactions*+)?
(**where** *postCondition*)? **.**

SPESC uses a natural-language-like concrete syntax to specify specific contract terms. Above figure demonstrates this claim where a term is associated to a party and actions they can take when a role (or condition) is specified.

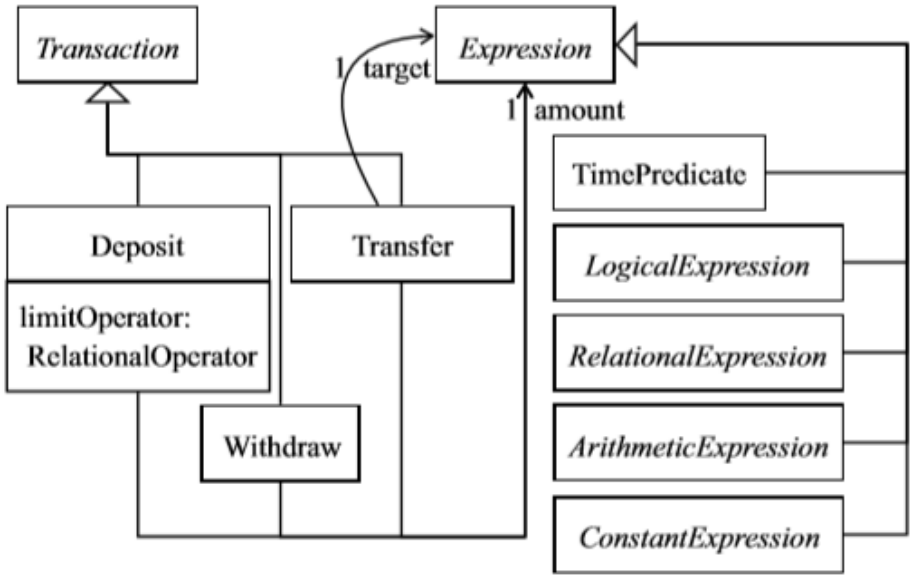


Diagram of expressions and transactions metamodel in SPESC. [29]

Expression in SPESC follows a syntax like- (all | somel this) ? party did action. On bases of a particular condition satisfying action- Transaction, Deposit, Transfer, Withdraw will take place. TimePredicates, LogicalExpression, RelationExpression, ArithematicExpression, ConstantExpression are defined predicates for an expression. If a party does not perform an action, query returns ⊥, where ⊥ denotes unsolved. Figure below demonstrates a concrete example of three transaction operation: [29]

deposit $(= \mid > \mid >= \mid < \mid <=)?$ **$**$amount$
withdraw **$**$amount$
transfer **$**$amount$ **to** $target$

Drawback - Although SPESC specialization is an amazing way to do collaborative work in designing smart contract, one still requires some basic knowledge of reading syntax - understand the format, grammar, and the keywords used in writing such contracts.

Now we can agree that smart contracts immutable nature refrain us from fixing vulnerability in existing smart contracts. While this is the truth, we must take care in development - using secure guidelines and with the help of other non-IT individuals expertise in contracts.

In conclusion, the above-mentioned ways mentioned in section 8 will help IT and non-IT developers (writing contracts in SPESC) write secure contracts and unleash the potential of digital smart-contracts in this growing internet industry. SPECS specialization is recommended for writing smart contracts in collaboration with other person or entity - it is easy to read, understand and write.

Using FsolidM framework, recommended for smart-contract designers/developers to design semantically secure smart contracts as Finite State Machine with an aim to keep in mind every possible action user using this smart contract can take and to handle them. It also provides a graphic UI to design these machines by designers and developers together to deliver very secure contracts.

Conclusions and Recommendations

Although a relatively new idea, Blockchain is enjoying a fast rate of innovation, with new platforms being developed, and many features being added. Many Blockchain platforms are accessible across the World, but there is not a single platform that meets all needs:

- **Ethereum:** built for restrained access versus mass consumption and using Proof of Work
- **Hyperledger Fabric:** utilizing a modular architecture to support permissioned networks
- **Hyperledger Sawtooth:** open source, enterprise grade using Proof of Elapsed Time)
- **Ripple:** on advanced, scalable, fast blockchain technology using probabilistic voting
- **Stellar:** similar to Ripple for finance, but languages not Turing complete
- **Cardano:** like Ethereum, offers security and scalability by way of layered architecture, uses proof of stake consensus mechanism, and has imperative languages available
- **Neo:** known as the "Ethereum of China" aimed at developing a smart contract platform that has all the benefits of an Ethereum Virtual Machine, without weakening developers with language barriers in that developers can use Java, C# or other languages
- **EOS:** open-source aimed at providing decentralized storage of enterprise solutions, solving the scalability issues of Ethereum and Bitcoin, and eliminating fees for users
- **Eris:** open source smart contract application platform using capabilities-based permissions using the Ethereum VM and know Serpent or Solidity

As a result, organizations wishing to use blockchain as a tool must undertake an analysis in order to determine the right platforms - particularly when high scalability of the application is a requirement.

A particularly useful blockchain-based option is a smart contract: computer program which digitally memorializes contractual terms and conditions negotiated by users, and calls for the triggering of terms - including those that require payment - when those required conditions are met. With the advance in blockchain technology, smart contracts are being used to serve a wide range of purposes.

Blockchain technology is growing in prevalence in the realm of computer science and applications, and it appears that smart contracts will be one of the early ways that blockchain technology will impact the general public. Now that more and more commerce is being undertaken online, between people and organizations in disparate locations, over publicly accessible channels, commerce can be accomplished more efficiently with smart contracts and their underlying blockchain technology platform.

Smart contracts use cases are not limited to the ones discussed in this Report. Today, numerous computer scientists and policy makers work on the projects aimed at solving the acutest problems with the help of the blockchain technology and smart contracts, such as:

- **Digital Identity:** details about your life
- **Banking:** transfer money with no or little fee
- **Tax Records:** organizing records
- **Insurance:** organization of policies
- **Real Estate and Land Titles Recording:** public records
- **Supply** Chain: food safety
- **IoT (Internet of Things):** reduction of friction in a smart house or factory
- **Authorship and Intellectual Property Rights:** maintaining a transparent registry
- **Life Science and Health Care:** on demand health records
- **Gaming and Gambling:** honest record-keeping
- **Educational Records:** permanent, immutable, searchable transcripts

But how do parties to smart contracts know they can rely on blockchain platforms and smart contracts? By using the well-proven power of advanced cryptography algorithms and their secure private keys, smart contracts can be relied upon today as properly signed by the signing party as the digital signature protocols are well-designed and proven resistant to attack.

On a more in-depth, computer science level, smart contracts on top of a blockchain platform can be formally verified. Modeling formulas have been applied, and have shown the general reliability of smart contracts. And that formal modeling can highlight issues facing developers writing smart contracts.

There are ways to assure the security of smart contracts. Certain tools, like the FsolidM framework, can assist smart-contract designers/developers to design semantically secure smart contracts as finite state machines with an aim to keep in mind every possible action user using this smart contract can take and to handle them. It also provides a graphic UI to design these machines by designers and developers together to deliver very secure contracts.

There are however issues of security that are raised by both blockchain and smart contracts.

Development platforms for **blockchain** are always evolving. This technology is still at a nascent stage. The developer community is trying to assure that blockchain meets its promise of secure transactions. But the correct framework on Ethereum and other similar frameworks, though secure enough to be used in

the real world with real transactions, do not have well-defined and well-documented standards to write code. This has led to many vulnerabilities in poorly written code leading to attacks. To make developers more confident in writing code for Blockchain, there is a need for standardization and documentation of the frameworks on which they run their programs.

Ethereum and other platforms' digital signature protocols rely on the secured privacy of the signer's private key, thus any breach of the security of these keys would allow malicious actors to undertake seemingly unassailable signatures on behalf of another party. This is the age-old problem of keeping something under lock and key safe from those who wish to take it.

It has also been rumored that the National Security Administration has inserted a backdoor of sorts in the elliptical curve cryptography protocol used by Ehtereum, which raises a whole group of concerns related to privacy and government interference in commerce, especially in totalitarian regimes.

Future technology such as quantum computing and its potential to unlock new avenues of brute-force and other calculations could eventually provide a crack in the security provided by cryptographic and digital signature protocols currently deemed secure.. Although cryptography has made great strides since RSA was developed in the 1970s, history has proven that no cryptographic solution is secure forever.

Simulation with the statistical model checking tools reveal scenarios where the blockchain can be breached by hackers:

Scenario 1: hacker retrieves the name from the pending transactions data when the user transaction is not mined yet. Here the hacker has an average of 12% to hack the register, succeeding when both the hacker and the user transactions are in the pending transaction list. Due to the random mining, the hacker transaction can be mined and executed first

Scenario 2: He gets the name from the network, that is, directly from the user call interaction with the blockchain. Statistical Model Checking (SMC) probability evaluation feature with parameters α = 0.1 and δ = 0.1 is used [21]. Here the hacker has an average of 25% to hack the register, succeeding by intercepting the transaction while the user sends the register call transaction.

Since in scenario 1, the hacker should wait for the user's transaction to be in the pending transaction list, it explains the smaller chance for the hacker to succeed than in scenario 2. In order to avoid such attacks, a rigorous registration process should require two steps. First the user registers the hash of the name and then only he registers the actual corresponding name in a second transaction.

Smart contracts have their imperfections too. The code for any smart contract is written by a programmer which implies there are always chances of errors. Therefore, the human factor can't be solved entirely and it is never possible to be sure that there are no mistakes in the contract. There is a lot of work ahead for blockchain engineers and smart contract developers.

Another major setback for current Blockchains is the computational need to participate in the network. To make Blockchain to be widely accepted, the main problems to be addressed are the robustness of the code, scalability issues and computational needs of the network. Till recently participating in a blockchain meant that a miner had to build complex computational systems with huge power demands, many GPUs, a server grade processor and network adaptor to handle the computational load to be part of a network and also be profitable in mining.

The main aim for the Blockchain community is to make mining accessible to the large masses so that many people can participate in the mining process. This will facilitate the distribution of load over a bigger network because of increased availability of processing devices. This would also mean that the network would comprise of relativity less powerful machines (like a laptop or home desktop), which are already in abundance, connected to the biggest network in the world - the World Wide Web. This would encourage more miners and ultimately help in reaching the goal of a blockchain network which is having a completely decentralized system to handle transactions, by cutting down the middle-men and creating complete transparency in all the transactions in the network.

Finally, the immutable nature of smart contracts prevents the remedy of vulnerability in existing smart contracts. This is why it is important to understand blockchain and smart contracts, and for engineers working on these technologies to focus on keeping the widespread, utilitarian technological aids secure from onset.

REFERENCES

[1] Maher Alharby, Aad van Moorsel. 2017. Blockchain-based Smart Contracts: A Systematic Mapping Study. In: Fourth Int'l Conf. on Computer Science and Information Technology (CSIT-2017), 2017.

[2] Marco Iansiti, Karim R. Lakhani. 2017. The Truth About Blockchain - Harvard Business Review. Retrieved from: https://hbr.org/2017/01/the-truth-about-blockchain

[3] Paul Baran. 1964. On Distributed Communications - Introduction to Distributed Communication Networks. United States Air Force under Project Rand - Contract No, AF 49(638)-700. August 1964.

[4] Andrew Tar. 2018. Proof-of-Work, Explained | Cointelegraph. January 17, 2018. Retrieved from: https://cointelegraph.com/explained/proof-of-work-explained

[5] A. Panarello, N. Tapas, G. Merlino, F. Longo, A Puliafito. Blockchain and IoT Integration: A Systematic Survey. 2018. Sensors (Basel). 2018 Aug; 18(8): 2575. Published online August 6, 2018.

[6] Toshendra Kumar Sharma. 2019. Best Programming Languages to Build Smart Contracts. (October 31, 2019). Retrieved November 4, 2019 from: https://www.blockchain-council.org/blockchain/best-programming-languages-to-build-smart-contracts/

[7] Akash Takyar. 2019. Top Blockchain Platforms of 2019 for Blockchain Application - LeewayHertz. (October 2019). Retrieved from: https://www.leewayhertz.com/blockchain-platforms-for-top-blockchain-companies/

[8] Anonymous. 2019. Smart Contract Platforms [A Deep Dive Investigation]. (October 2019). Retrieved November 4, 2019 from https://blockgeeks.com/guides/different-smart-contract-platforms/#Stellar

[9] M. Macdonald, Lisa Liu-Thorrold, R. Julien. 2017. The Blockchain: A Comparison of Platforms and Their Uses Beyond Bitcoin. 10.13140/RG.2.2.23274.52164.

[10] N. Szabo. Formalizing and securing relationships on public networks. First Monday, 2(9), 1997.

[11] Yining Hu, Madhusanka Liyanage, Ahsan Mansoor, Kanchana Thilakarathna, Guillaume Jourjon, Aruna Seneviratne. June 8, 2019. Blockchain-based Smart Contracts - Applications and Challenges

[12] Petar Tsankov, Andrei Dan, Dana Draschler-Cohen, et al. 2018. Securify: Practical security analysis of smart contracts. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, Canada - October 15-19, 2018 pages 67-82

[13] Shuvenda K. Lahiri, Shuo Chen, Yeupeng Wang, Isil Dillig, et al. (2018) Formal Specification and Verification of Smart Contracts for Azure Blockchain. In: arXiv:1812.08829v2 [cs.PL]

[14] Anastasia Mavridou, Aron Laszka. 2018. Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach. In: Meiklejohn S., Sako K. (eds) Financial Cryptography and Data Security. FC 2018. Lecture Notes in Computer Science, vol 10957. Springer, Berlin, Heidelberg

[15] Jonathan Katz. 2010. Digital Signatures. Springer, New York, NY.

[16] Bruce Schneier. 1996. Applied Cryptography. (2nd. ed.) Wiley, Indianapolis, IN.

[17] Whitfield Diffie, and Martin Hellman. (1976). New directions in cryptography. IEEE Trans. Inform. Theory IT-22, 6 (Nov. 1976), 644-654.

[18] R. L. Rivest, A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 2 (February 1978), 120-126.

[19] Anonymous. Cryptography Digital Signatures. Retrieved from: https://www.tutorialspoint.com/cryptography/cryptography_digital_signatures.htm

[20] Dominiek Ter Heide. A Closer Look at Ethereum Signatures. Retrieved from: https://hackernoon.com/a-closer-look-at-ethereum-signatures-5784c14abecc

[21] Tesnim Abdellatif, Kei-Leo Brousmiche. 2018. Formal Verification of Smart Contracts Based on Users and Blockchain Behavior Models. In: 2018 9th IFIP Int'l Conf. on New Technologies, Mobility and Security (NTMS), February 26-28, 2018

[22] Alyssa Hertig. How do Ethereum Smart Contracts Work. Retrieved from: https://www.coindesk.com/information/ethereum-smart-contracts-work

[23] Anonymous. Smart Contracts: 10 Use Cases for Business. Retrieved from: https://ambisafe.com/blog/smart-contracts-10-use-cases-business/

[24] Danda B. Rawat, Vijay Chaudhary, Ronald Doku. 2019. Blockchain: Emerging Applications and Use Cases. In: arXiv:1904.12247vl [cs:CR]

[25] Iuon-Change Lin, Tzu-Chun Liao. 2017. A survey of blockchain security issues and challenges. International Journal of Network Security. 19. 653-659..

[26] Anonymous. 2019. Ethereum Smart Contract Best Practices. Retrieved from: https://consensys.github.io/smart-contract-best-practices/known_attacks/

[27] Naoki Shibata, August 6th, 2019. Blockchain Consensus Formation while Solving Optimization Problems. Retrieved from: http://arxiv.org/abs/1908.01915v1

[28] Anonymous. Know attacks in smart contracts. Retrieved from (deep link to [26]): https://consensys.github.io/smart-contract-best-practices/known_attacks/

[29] He, Xiao, Bohan Qin, Yan Zhu, Xing Chen, and Yi Liu. 2018. SPESC: A specification language for smart contracts. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), vol. 1, pp. 132-137. IEEE, 2018.