Smart Contracts using Blockchain

Rajat Rawat[1], Rohan Chougule[2], Shivam Singh[3], Shiwam Dixit[4], GUIDED BY – Prof. Anup Kadam[5]

[1,2,3,4,5]*Computer Engineering Department, Army institute of Technology, Pune*

***Abstract*—**In recent years, the rapid development of cryptocur- rencies and their underlying blockchain technology has revived Szabo's original idea of smart contracts, i.e., computer protocols that are designed to automatically facilitate, verify, and enforce the negotiation and implementation of digital contracts without central authorities. Smart contracts can find a wide spectrum of potential application scenarios in the digital economy and intelligent industries, including financial services, management, healthcare, and Internet of Things, among others, and also have been integrated into the mainstream blockchain-based develop-ment platforms, such as Ethereum and Hyperledger. However, smart contracts are still far from mature, and major technical challenges such as security and privacy issues are still awaiting further research efforts. For instance, the most notorious case might be "The DAO Attack" in June 2016, which led to more than $50 million Ether transferred into an adversary's account. In this paper, we strive to present a systematic and compre-hensive overview of blockchain-enabled smart contracts, aiming at stimulating further research toward this emerging research area. We first introduced the operating mechanism and main- stream platforms of blockchain-enabled smart contracts, and proposed a research framework for smart contracts based on a novel six-layer architecture. Second, both the technical and legal challenges, as well as the recent research progresses, are listed. Third, we presented several typical application scenarios. Toward the end, we discussed the future development trends of smart contracts. This paper is aimed at providing helpful guidance and reference for future research efforts.

***Index Terms*—Blockchain, parallel blockchain, six-layer archi- tecture, smart contracts.**

INTRODUCTION

THE TERM "smart contract" was first coined in mid- 1990s by computer scientist and cryptographer Szabo, who defined a smart contract as "a set of promises, specified in digital form, including protocols within which the parties perform on these promises [1]." In his famous example, Szabo analogized smart contracts to vending machines: machines take in coins, and via a simple mechanism (e.g., finite automata), dispense change and product according to the displayed price. Smart contracts go beyond the vending machine by proposing to embed contracts in all sorts of properties by digital means [2]. Szabo also expected that through clear logic, verification and enforcement of cryptographic protocols, smart contracts could be far more functional than their inanimate paper-based ancestors. However, the idea of smart contracts did not see the light till the emergence of blockchain technology, in which the public and append-only distributed ledger technology (DLT) and the consensus mechanism make it possible to implement smart contract in its true sense.

Generally speaking, smart contracts can be defined as the computer protocols that digitally facilitate, verify, and enforce the contracts made between two or more parties on blockchain. As smart contracts are typically deployed on and secured by blockchain, they have some unique characteristics. First, the program code of a smart contract will be recorded and verified on blockchain, thus making the contract tamper-resistant. Second, the execution of a smart contract is enforced among anonymous, trustless individual nodes without centralized con- trol, and coordination of third-party authorities. Third, a smart contract, like an intelligent agent, might have its own cryp- tocurrencies or other digital assets, and transfer them when predefined conditions are triggered [3].

It is worth noting that Bitcoin[1] is widely recognized as the first cryptocurrency that support basic smart contracts, in the sense that its transactions will be validated only if certain con- ditions are satisfied. However, designing smart contract with complex logic is not possible due to the limitations of Bitcoin scripting language that only features some basic arithmetic, logical, and crypto operations.

[1]Bitcoin. https://bitcoin.org/.

Ethereum[2] is the first public blockchain platform that supports advanced and customized smart contracts with the help of Turing-complete virtual machine called Ethereum vir- tual machine (EVM). EVM is the runtime environment for smart contracts, and every node in the Ethereum network runs an EVM implementation and executes the same instructions. Several high-level programming languages, such as Solidity[3] and Serpent,[4] can be used to write Ethereum smart contracts, and the contract code is compiled down to EVM bytecode and deployed on the blockchain for execution. Ethereum is currently the most popular development plat- form for smart contracts, and can be used to design various kinds of decentralized applications (DApps), e.g., digital rights management, crowdfunding, gambling, etc.

Although smart contracts have made great progresses in recent years, it still faces many challenges. A well-known event is that in June 2016, The DAO, a decentralized investor- directed venture capital fund secured by Ethereum blockchain, was attacked by exploiting a severe smart contract bug called "Recursive call." The attacker drained more than $50 million Ether into a "child DAO" that has the same structure as The DAO. At last, a hard fork of the Ethereum was implemented to claw back the funds from the attacker. However, this hard fork was controversial because it violates the *code is law* principle in the spirit of blockchain technology. In addition to the secu- rity problem, other challenges include performance, privacy, legal issues, etc.

The main aim of this paper is to offer a comprehensive overview of smart contract research, including the operating mechanism, basic framework, application scenarios, challenges, recent progresses, future trends, etc.

The rest of this paper is organized as follows. Section II systematically introduces the smart contracts, including the operating mechanism and mainstream development platforms, and a basic research framework which employs a six-layer architecture is proposed. Section III summarizes the current challenges faced by smart contracts and the recent research progresses. Section IV presents several typical application sce- narios of smart contracts, e.g., finance, management, Internet of Things (IoT), and energy. Section V discusses the future development trends. Section VI concludes this paper.

SMART CONTRACTS

In this section, we will give an overview of smart contracts. First, we make a brief introduction to blockchain, and then present the operational mechanism of smart contracts based on two mainstream platforms—Ethereum and Hyperledger Fabric. We also propose a basic research framework of smart contracts.

A. *Brief Introduction to Blockchain*

The concept of blockchain originated from Bitcoin, which is a cryptocurrency invented by an unknown people or group of people using the pseudonym Nakamoto in 2008 [4].

[2]Ethereum. https://www.ethereum.org/.

[3]Solidity. http://solidity.readthedocs.io/en/latest/.

[4]Serpent. https://github.com/ethereum/wiki/wiki/Serpent.

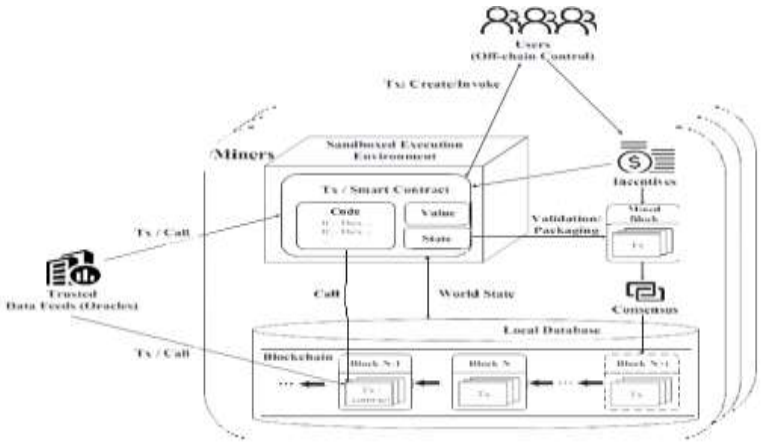

Fig. 1. Operational mechanism of smart contract.

Blockchain is a continuously growing list of records, called blocks, which are linked and secured using cryptography. Blockchain adopts the P2P protocol that can tolerate single point of failure. The consensus mechanism ensures a common, unambiguous ordering of transactions and blocks, and guaran- tees the integrity and consistency of the blockchain across geographically distributed nodes. By design, blockchain has such characteristics as decentralization, integrity, and auditability [5]. According to Xu *et al.* [6], blockchain can serve as a novel kind of software connector, which should be considered as a possible decentralized alternative to the existing centralized shared data storage. In addition, based on different levels of access permission, blockchains can be divided into three types: 1) public blockchain (such as Bitcoin and Ethereum);

2) consortium blockchain (such as Hyperledger[5] and Ripple);[6] and 3) private blockchain. Blockchain serves as the platform for smart contracts to be hosted and executed on.

Smart contracts are introduced as computer programs run-ning across the blockchain network and can express triggers, conditions, and business logic to enable complicatedly pro- grammable transactions [6]. In the next section, we will discuss the operational mechanism of smart contracts in detail.

B. Operational Mechanism of Smart Contracts

The operational mechanism of smart contracts is shown in Fig. 1. Smart contracts generally have two attributes: 1) value and 2) state. The triggering conditions and the corresponding response actions of the contract terms are preset using triggering condition statements such as "If-Then" statements. Smart contracts are agreed upon and signed by all parties and submitted in transactions to the blockchain network, then transactions are broadcasted via the P2P network, verified by

[5]Hyperledger. https://www.hyperledger.org/.

[6]Ripple. https://ripple.com/.

SMART CONTRACTS USING BLOCKCHAIN

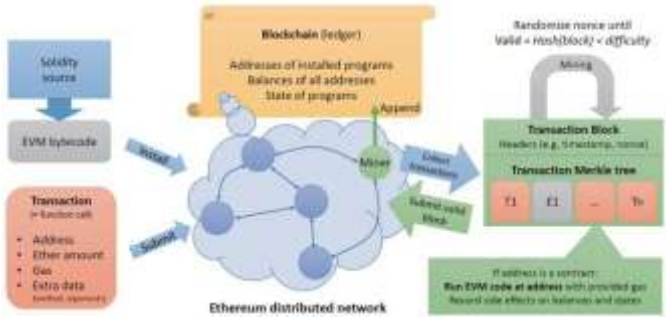

Fig. 2. Overview of workflow in the Ethereum network [8].

The miners and stored in the specific block of the blockchain. The creators of the contracts get the returned parameters (e.g., contract address), then users can invoke a contract by sending a transaction. Miners are motivated by the system's incentive mechanism and will contribute their computing resources to verify the transaction. More specially, after the miners receive the contract creation or invoking transaction, they create contract or execute contract code in their local Sandboxed Execution Environment [(SEE), e.g., EVM]. Based on the input of trusted data feeds (also known as, Oracles) and the system state, the contract determines whether the current scenario meets the triggering conditions. If the conditions are met, the response actions are strictly executed. After a transaction is validated, it is packaged into a new block. The new block is chained into the blockchain once the whole network reaches a consensus.

Next, we take Ethereum and Hyperledger Fabric as exam- ples to introduce the operational process of smart contracts.

1) Ethereum: Ethereum is currently the most widely used smart contracts development platform that can be viewed as a transaction-based state machine: it begins with a genesis states and incrementally executes transactions to morph it into some final states. It is the final states which we accept as the canonical "version" in the world of Ethereum [7]. Unlike the UTXO model of Bitcoin, Ethereum introduces the concept of accounts. There are two types of accounts: 1) externally owned accounts (EOAs) and 2) contract accounts. The difference is that the former is controlled by private keys without code asso-

ciated with them, while the latter is controlled by their contract code with associated code.

Users can only initiate a transaction through an EOA. The transaction can include binary data (payload) and Ether. If the recipient of a transaction is the zero-account, a smart contract is created. Or if the recipient is a contract account, the account will be activated and its associated code is executed in the local EVM (the payload is provided as input data) [8]. The transaction is then broadcast to the blockchain network where miners will verify it, as shown in Fig. 2.

In order to avoid issues of network abuse and to sidestep the inevitable problems stemming from Turing completeness, all programmable computations (e.g., creating contracts, mak- ing message calls, utilizing and accessing account storage, and executing operations in the virtual machine) in Ethereum is subject to fees—a reward for miners who contribute their

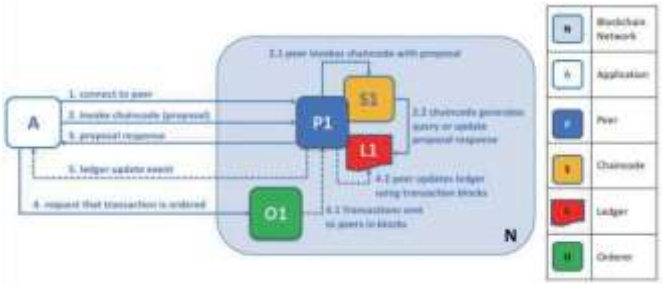

Fig. 3. Transaction workflow of Hyperledger Fabric.[7]

computing resources. The unit used to measure the fees required for the computations is called gas [7].

2) Hyperledger Fabric: Hyperledger Fabric[8] is a block- chain framework implementation and one of the Hyperledger projects hosted by The Linux Foundation. Rather than the pub- lic blockchain, such as Bitcoin and Ethereum that anybody can participate in the network, Hyperledger Fabric is permissioned because only a collection of business-related organizations can join in through a membership service provider, and its network is built up from the peers who are owned and contributed by those organizations. Peers are hosts for ledgers and chaincodes (smart contracts). The ledger is the sequenced, tamper-resistant record of transactions/state transitions. State transition is a result of chaincode invocation (transaction). Each transaction results in a set of asset key–value pairs that are committed to the ledger as creates, updates, or deletes. As shown in Fig. 3, the transaction workflow of Hyperledger Fabric consists of three phases as follows.

1) *Proposal:* An application sends a transaction proposal to different organizations' endorsing peers (also called endorsers who validate transactions against endorsement policies and enforce the policies). The proposal is a request to invoke a chaincode function so that data can be read and/or written to the ledger. The transaction results include a response value, read set, and write set. The set of these values, along with the endorsers' sig- natures are returned to the application as a transaction proposal response.

2) *Packaging:* The application verifies the endorsers' signa- tures and checks if the proposal responses are the same. Then, the application submits the transaction to order- ing service (orderer) to update the ledger. The orderer sorts the transactions it received from the network, and packages batches of transactions into a block that ready for distribution back to all peers connected to it.

3) *Validation:* The peers connected to the orderer vali- date every transaction within the block to ensure that it has been consistently endorsed by required organiza- tions according to the endorsement policy. It is worth noting that this phase does not require the running of chaincode—this is only done in proposal phase. After

[7]Hyperledger Fabric Docs. http://hyperledger-fabric.readthedocs.io/en/ release-1.1/peers/peers.html.

[8]Hyperledger Fabric. https://www.hyperledger.org/projects/fabric. validation, each peer appends the block to the chain, and the ledger is updated.

Ethereum and Hyperledger Fabric differ in the following aspects. First, Ethereum is a public blockchain platform, while Hyperledger Fabric is a consortium blockchain infrastruc- ture in that only a predefined community of participants are permissioned to join the network. Comparatively speaking, Hyperledger Fabric has high degrees of scalability, resilience, and

confidentiality as it provides a modular architecture with a delineation of roles between the nodes (e.g., endorsers and orderers) and configurable consensus and membership ser- vices. Second, in Hyperledger Fabric, there is no built-in cryptocurrency or fuel (such as Ether and gas in Ethereum). Third, the chaincode in Hyperledger only defines a set of assets which are presented as key–value pairs, and provides the functions for operating on the assets and changing their states. Last, for contract code execution, the contract code in Ethereum is included in a transaction which is propagated in the P2P network, and any miner who receives this transaction can execute it in their local virtual machine. However, in Hyperledger Fabric, the chaincode is actually hosted by peer nodes (peers). When a transaction is created by the application, the transaction is only executed and signed by specified peers (endorsing peers). After receiving the application's transaction proposal, each of these endorsing peers independently executes it by invoking the chaincode to which the transaction refers. For security, chaincode runs within a container environment (e.g., Docker) for isolation.

It is worth mentioning that the intersection between Ethereum and Hyperledger is widening. For instance, the Hyperledger Burrow project that runs under Tendermint con- sensus engine has begun to support running Ethereum smart contracts on Fabric using Hyperledger Fabric EVM chaincode plugin.

C. Basic Research Framework of Smart Contracts

According to the operational mechanism of smart con- tracts, we summarize the life-cycle of a smart contract into five stages: 1) negotiation; 2) development; 3) deployment;

4) Maintenance; and 5) learning and self-destruction. Based on this life-cycle, we propose a basic research framework of smart contracts. The framework also refers to several previous literatures. For example, Risius and Spohrer [9] presented a research framework to structure the insights of the current body of research on blockchain technology. Xu *et al.* [10] proposed a taxonomy to classify and compare blockchains and blockchain-based systems. The taxonomy captures major architectural characteristics of blockchains and the impact of different design decisions, which helps with important architectural considerations about the performance and quality attributes of blockchain-based systems [10]. Glaser [11] devel- oped a comprehensive conceptual framework of blockchain systems and further divided blockchain systems into two layers of code, namely, fabric layer and application layer.

As shown in Fig. 4, the proposed research framework employs a six-layer architecture, namely, infrastructures layer, contracts layer, operations layer, intelligence layer,

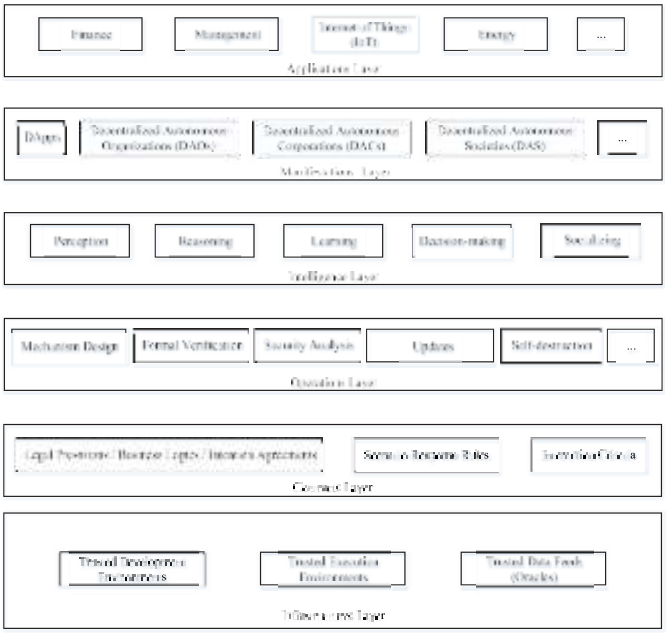

Fig. 4. Basic research framework of smart contracts.

Manifestations layer, and applications layer from the bottom up. The details are as follows.

1) *Infrastructures Layer:* The infrastructures layer encapsu- lates all the infrastructures that supports smart contracts and

their applications, including the trusted develop- ment environments, trusted execution environments, and trusted data feeds (Oracles). To a certain extent, the choice of these infrastructures will affect smart con- tracts' design patterns and contract attributes.

a) *Trusted Development Environments:* In the pro- cess of smart contracts development, deployment, and invoking, a variety of development tools are involved, e.g., programming languages, integrated development environments (IDEs), development frameworks, clients, wallets, etc. Taking the wal- let as an example, in addition to being a digital asset management tool, it usually assumes func- tions, such as being a boot node, deploying a contract, and invoking a contract.

b) *Trusted Execution Environments:* Blockchain pro- vides the trusted execution environment for smart contracts. The execution of the smart contracts rely on blockchain's key components, such as con- sensus algorithm, incentive mechanism, and P2P network, and the final execution results will be recorded in the distributed ledger maintained by all nodes. Different consensus algorithms and incen- tive mechanisms will affect the design pattern, exe- cution efficiency and security of smart contracts. For example, the development and deployment of smart contracts in Ethereum must consider the fuel consumption to avoid denial-of-service attacks and unnecessarily high costs caused by massive call- ing of dead code, opaque predicates, and expensive operations in a loop and other gas-costly opera- tions, as well as the out of gas exception caused by gas shortage.

SMART CONTRACTS USING BLOCKCHAIN

c) *Trusted Data Feeds (Oracles):* In order to guar- antee the security of blockchain network, smart contracts are generally executed in SEE (e.g., EVM in Ethereum and Docker container in Hyperledger Fabric), which is not allowed to import external information. Hence, smart contract needs trusted data feeds (Oracles) to provide external states about the real world in the form of a transaction (because any information that is not generated by a trans- action has to be introduced as data attached to a transaction [11]) in a secure and trusted man- ner, thereby ensuring the deterministic of contract execution results.

2) *Contracts Layer:* The contracts layer encapsulates the static contract data, including contract terms, scenario- response rules, and interaction criteria. Thus, this layer can be regarded as the static database of smart contracts which includes all the rules about contract invocation, execution, and communication. When a smart contract is being designed, at first all parties (contractors) shall negotiate and determine the contract terms which may involve legal provisions, business logics, and intention agreements. Then, programmers use software engineer- ing technology, such as algorithm design and design pat- tern to translate the contract terms described in natural language into the program code, e.g., a series of If-Then- type scenario-response rules. Moreover, interaction cri- teria (e.g., access authority, communication mode, etc.) should also be enacted in this layer for the interactions between contracts and users (or contracts and contracts) according to the characteristics of the development platforms and contractors' intentions.

3) *Operations Layer:* The operations layer encapsulates all the dynamic operations on the static contracts, including mechanism design, formal verification, security analy- sis, updates, and self-destruction. Maintenance layer is the key to the correct, safe, and efficient operation of smart contracts because malicious or vulnerable smart contracts can bring huge economic losses to users. From the perspective of smart contracts' life-cycle from nego- tiation to self-destruction, before the smart contracts are deployed onto the blockchain, mechanism design operations use information and incentive theory to help contracts achieve their function efficiently. Formal ver- ification [8] and security analysis operations are used to verify the correctness and security of contract codes, and ensure that the codes will be executed according to the programmers' actual semantics [12]. After the smart contracts are deployed onto the blockchain, updates can be implemented technically when the contract func- tion is difficult to meet users' demands or the contract has repairable vulnerabilities, although all the historical updates are recorded on the blockchain and cannot be tampered. At the end of the smart contracts' life cycle or when a high-risk vulnerability occurs, self-destruction is conducted to insure network security.

4) *Intelligence Layer:* The intelligence layer encapsu- lates various intelligent algorithm, including perception, reasoning, learning, decision-making, and socializing, which add intelligence to the smart contracts built on the first three layers. It must be pointed out that current smart contracts do not have much intelligence. However, we believe that the future smart contracts will not only be self-enforcing according to the predefined If-Then statements but also should have "What-If"-type deduc- tion, computation, and intelligent decision-making in unknown scenarios. As mentioned earlier, smart con- tracts running on the blockchain network can be consid- ered as software agents that act on behalf of their users. With the development of artificial intelligence (AI) tech- nology, agents will have a certain degree of intelligence,

such as perception, reasoning, and learning by virtue of cognitive computing [13], reinforcement learning [14], etc. Hence, those agents are not only autonomous as they have capabilities of tasks selection, prioritization, and goal-directed behaviors (sometimes referred to as belief- desire-intention [15]) but also have sociability through communication, cooperation, and negotiation with each other. The learning and collaboration results will also be fed back to the contracts layer and the operations layer, thus optimizing the contract design and operation, ultimately realizing the truly "smart" contract.

5) *Manifestations Layer:* The manifestations layer encap- sulates various manifestation forms of smart contracts for potential applications, including DApps, decentral- ized autonomous organizations (DAOs), decentralized autonomous corporations (DACs), and decentralized autonomous societies (DASs). Smart contracts that encapsulate the complex behaviors of network nodes are equivalent to the application interfaces of blockchain, which enable blockchain to embed different applica- tion scenarios. For instance, by writing legal provi- sions, business logics, and intention agreements into smart contracts, a variety of DApps can be devel- oped. Furthermore, the multiagent systems built on the fourth layers will gradually evolve into various DAOs, DACs, and DASs. These high-level manifesta- tion forms are expected to improve traditional business and management, and lay the foundation for the future programmable society. Taking the DAO as an exam- ple, DAOs are organizations that are powered and run by smart contracts, their business and adminis- trative rules are all recorded on blockchains. DAOs can reduce transaction costs and introduce the possi- bility of aligning interests for stakeholders in a more decentralized manner. Therefore, DAOs are expected to bring disruptive influence to the traditional management paradigms which are typically in a top-down hierarchical structure [16].

6) *Applications Layer:* The applications layer encapsu- lates all the application domains that built upon the manifestation layer. For instance, based on DAO, an application called Plantoid (also named as the distributed autonomous art) was developed in Ethereum, which realized a truly aesthetic economy that binds artists, designers, artworks, and audiences into a symbiotic relationship, thereby emancipating art from concentrated and hierarchically organized capitalist markets [17]. Theoretically, smart contracts can be used in all indus- tries, e.g., finance, IoT, healthcare, supply chain, etc. We will introduce them in detail in Section IV.

It is worth noting that the proposed framework is in only an ideal framework, especially for the intelligence layer. However, just as pointed out by Glaser [11], it is a functional limitation that any activity in the blockchain needs to be trig- gered by a node controlled from outside of the network, and smart contract should implement autonomous mechanisms or complex microservice interactions which, in total, realize more sophisticated service logic like an autonomous portfolio man- agement service. Future smart contracts should have a certain autonomy and intelligence.

The proposed framework is of a certain theoretical and prac- tical value for researchers and practitioners. On the one hand, the framework covers the key elements in the whole life-cycle of smart contract. On the other hand, the framework indicates the research direction and possible development trends.

CHALLENGES AND RECENT PROGRESSES

As an emerging technology in its infancy, smart contracts currently face many problems and challenges. Based on the proposed research framework which employs a six-layer archi- tecture, this section will outline the challenges and recent research progresses of smart contracts.

A. Contract Vulnerabilities

Contract vulnerabilities mainly appear in the contracts layer in the research framework we proposed. The malicious miners or users can exploit them to gain profit. Here are some typical cases [18]–[20].

1) *Transaction-Ordering Dependence (TOD):* Each block contains several transactions, and the order in which transactions are executed depends on the miner. TOD occurs when several dependent transactions invoke the same contract that the miner can manipulate the order in which the transactions are executed.

2) *Timestamp Dependence:* The miners set the timestamp for the block they mined (generally according to the miner's local clock system). The miner can modify the timestamp by a few seconds on the promise that other miners accept the block they proposed. The vulnerability lies in the fact that some smart contracts take timestamp as a trigger condition, e.g., transferring money, thus adversary may manipulate the timestamp-dependent contracts for their own interests.

3) *Mishandled Exceptions:* When a contract (caller) calls another contract (callee), if the callee runs abnormally, it terminates and returns false. This exception may or may not be passed to the caller. In principle, the caller must explicitly check the return value from the callee to verify that the call was executed successfully. However, If the caller does not properly check the return value, it will bring potential threats. A typical case is the King of the Ether Throne contract in Ethereum.

To deal with those contract vulnerabilities, some secu- rity analysis tools are developed. For example, as many contract bugs stem from a semantic gap between the programmers about the underlying execution semantics and the actual semantics of the smart contracts, Luu et al. [12] developed a symbolic execution tool called Oyente to find potential security bugs in Ethereum smart contracts. Among 19 366 smart contracts in Ethereum, Oyente flagged 8833 of them as vulnerable, including The DAO bug [12]. Securify [23] is a security analyzer for Ethereum smart contracts. Its analysis consists of two steps: first, it symbolically analyzes the contract's depen- dency graph to extract precise semantic information from the code. Then, it checks compliance and violation patterns that capture sufficient conditions for proving if a property holds or not. Securify can analyze many vulnerabilities, such as transaction-reordering, recursive calls, insecure coding pat- terns, etc. Manticore9 is another symbolic execution tool for analysis of binaries and smart contracts which can record an instruction-level trace of execution for each generated input and discover inputs that crash programs via memory safety violations. Remix10 is a web-based IDE which serves as a security tool by analyzing the Solidity code to reduce coding mistakes and identify potential vulnerable coding patterns.

B. Limitations of the Blockchain

The limitations in blockchain itself are important factors hindering the development of smart contract. These limita- tions correspond to the infrastructures layer of the smart contract framework we proposed. Some typical limitations are as follows.

1) *Irreversible Bugs:* Due to the irreversible nature of the blockchain, once the smart contracts are deployed, they are finalized and cannot be changed. In other words, if there exists a bug in a smart contract, there is no direct way to fix it. Thus, if you find a defect in a smart contract, you need to update it. And when you deploy a new version of an existing contract, data stored in the

[9]Manticore. https://github.com/trailofbits/manticore.

[10]Remix. https://github.com/ethereum/remix.

SMART CONTRACTS USING BLOCKCHAIN

Previous contract is not automatically transferred—you have to manually initialize the new contract with the past data which makes it very cumbersome.

2) *Performance Issues:* Performance issues in blockchain systems, such as limited scalability, throughput bottle- neck, transactions latency, and storage constraints also limit the performance of smart contracts. Taking the throughput as an example, in the current blockchain systems, smart contracts are executed serially by miners and validators. Serial execution limits system through- put and fails to exploit today's concurrent multicore and cluster architectures. Dickerson *et al.* presented a novel way to permit miners and validators to execute smart contracts in parallel, based on techniques adapted from software transactional memory.

3) *Lack of Trusted Data Feeds (Oracles):* As mentioned before, the execution of smart contract requires the exter- nal data about real-world states and events from outside the blockchain, trusted data feeds (Oracles) serve as the bridge between blockchain and the external world (e.g., Web API). Lacking a substantive ecosystem of trustworthy data feeds is often regarded as a critical obstacle to the evolution of smart contracts. For this problem, we found a town crier (TC) solution that acts as a reliable connection between HTTPS-enabled websites and blockchain to provide authenticated data feeds for smart contracts. Oraclize[11] is an Oracle service for smart contracts and blockchain applications, which guarantees the data fetched from the original data-source is genuine and untampered by accompanying the returned data together with a document called authenticity proof. In addition, some prediction market platforms, such as Augur[12] and Gnosis[13] can also serve as Oracles, as they can pro- vide external information for smart contracts, such as the results of sports events or political elections.

4) *Lack of Standards and Regulations:* One of the pri- mary blockchain security issues and risks is the lack of standards and regulations. We proposed the concept of criminal smart contracts (CSCs), and listed some typical CSCs, e.g., leakage of confidential information, theft of cryptographic keys, and various real-world crimes (murder, arson, and terrorism). When malicious behaviors occur in smart contracts, it is difficult to supervise these malicious acts due to lack of effective regulation mechanism. In face of the potential high security risks of blockchain and smart contracts, some regulatory authorities, e.g., U.S. Securities and Exchange Commission began to pay attention to the regulatory and operational challenges arising from these new technologies .

C. *Privacy and Legal Issues*

The privacy issues of smart contracts can be divided into two categories: 1) contract data privacy and 2) trusted data

[11]Oraclize. http://www.oraclize.it/.

[12]Augur. https://www.augur.net/.

[13]Gnosis. https://gnosis.pm/.

Feeds privacy, involving the infrastructures layer and contracts layer of the research framework we proposed. Currently, not only transactions but also contract-related information are pub- licly available (especially for the information on the public blockchain), such as the bytecode, invoking param- eters, etc. So it represents a real challenge to keep critical functions/methods secret, apply cryptography, and avoid dis- closing data that should not have been public. Kosba *et al.* proposed a decentralized smart contract system called Hawk that allows developers to write privacy-preserving smart contracts without the need of implementing any cryptography, and its compiler automatically generates an efficient cryptographic protocol where contractual parties interact with the blockchain, using cryptographic primitives such as zero-knowledge proofs. Watanabe *et al.* proposed to encrypt smart contracts before deploying them on the blockchain. Only those participants who involved in a contract can access its content by using the decryption keys . For trusted data feeds privacy, TC supported private and custom data requests, enabling encrypted requests and secure use of access-controlled, off-chain data sources.

The legal issues of smart contract are mainly embodied in the contracts layer. Some scholars argue that smart con- tract is merely a type of computer code that can self-enforce, self-verify, and self-constrain the performance of its instruc- tions, which may represent all, part, or none of a valid legal contracts under the existing laws. Hence, there may be a con- flict between relational contract theories and smart contracts. For example, data privacy laws in European stipulate that cit- izens have a "right to be forgotten" which is incompatible with the immutable nature of blockchain-enabled smart con- tracts. Other legal issues include, but are not limited to, the following.

1) What laws otherwise apply to the transactions taking place within the smart contract application?

2) What hazards are posed by use of the smart contract application alone (e.g., a) a loss of data; b) business interruption; c) privacy breach; and/or d) a failure to perform)?

3) What happens when the outcomes of a smart contract diverge from the outcomes that the law demands ?

In addition to the above challenges on infrastructures layer and contracts layer, there are some other challenges. For example, on operations layer, poor mechanism design of smart contracts will increase contracts execution costs and reduce contracts execution efficiency. Designers need to design a set of incentive mechanisms to align the individual interests with the overall interests of the organization/society, thus to achieve incentive compatibility. On intelligence layer, the malicious intelligent agents may profit from their malicious behavior, etc.

APPLICATION SCENARIOS OF SMART CONTRACTS

Currently, applications of smart contracts are springing up. This section will take finance, management, IoT, and energy as examples to introduce the application scenarios of smart contracts.

A. *Finance*

Blockchain and smart contracts enable increased visibility and trust across the participants while bring huge savings in

infrastructures, transactions, and administrative costs. The following are several typical applications of smart contracts in finance.

1) *Securities:* Security industry involves complex proce- dures that are time consuming, cost inefficient, cum- bersome, and prone to risks. Smart contracts can circumvent intermediaries in the chain of securities custody and facilitate the automatic payment of div- idends, stock splits, and liability management, while reducing operational risks. In addition, smart con- tracts can facilitate the clearing and settlement of securities. At present, major markets in the U.S., Canada, and Japan still have a 3-day settlement cycle (T 3) that involves many institutions, such as securities depositories and collateral management agen- cies. The centralized clearing entails labor-intensive activities and complex internal and external reconcili- ations. Blockchain enables bilateral peer-to-peer execu- tion of clearing business logic using smart contracts. The Australian Securities Exchange is working on a DLT- based post-trade platform to replace its equity settlement system [38].

2) *Insurances:* The insurance industry spends tens of mil- lions of dollars each year on processing claims and loses millions of dollars to fraudulent claims. Smart contracts can be exploited to automate claims processing, verifi- cation, and payment, thus to increase the speed of claim processing as well as to eliminate fraud and prevent potential pitfalls. For example, The French airline, AXA,[14] is taking flight insurance to the smart con- tracts. If passengers' flight is more than two hours late, they will get automatically notified with the compensa- tion options. Smart contracts may also be used in auto insurance, because contracts can record the insurance clauses, driving records, and accident reports, allowing IoT-equipped vehicles to execute claims shortly after an accident.

3) *Trade Finance:* Trade finance is currently full of inef- ficiencies and the industry is extremely vulnerable to fraud. Besides, the paper-based processes of trade finance desperately need to be upgraded or replaced with digitalized operations. Smart contracts allow businesses to automatically trigger commercial actions based on predefined criteria that will boost efficiency by stream- lining processes, and reduce both fraud and compliance costs. In July 2017, a trade transaction was completed between Australia and Japan. This trade transaction saw all the trade-related processes, from issuing a letter of credit to delivering trade documents completed entirely via the Hyperledger Fabric platform, which reduced the time required to transmit documents, as well as the labor and other costs.

AXA. https://fizzy.axa/en-gb/.

B. Management

Blockchain-enabled smart contracts can provide appropriate and transparent accountability in terms of roles, responsibili- ties, and decision processes in management. Some use cases follow.

1) *Digital Properties and Rights Management:* Storing cryptographic certification of properties or rights on blockchain can facilitate the access and validation. de la Rosa *et al.* proposed to use smart contracts to certify the proof of existence and authorship of intel- lectual properties. Propy[15] allows owners and brokers to register their real estate properties, where buyers can search and negotiate the sale. Both parties participate in the smart contracts together and specific steps are taken throughout the process to ensure fair and legal play. Smart contracts can also be applied in digital rights management. For example, a DApp called Ujo Music[16] enforce the royalty payments for a musician once his/her work is used for commercial purposes.

2) *Organizational Management:* Now, most organizations are managed by and centered on a board of direc- tors who hold majority of decision-making power. It is believed that the future organizational management will be flattened and decentralized. Smart contracts can remove unnecessary intermediaries that impose artificial restrictions and unnecessarily complex regulations. For example, Aragon[17] is a project powered by Ethereum that aims to disintermediate the creation and mainte- nance of organizational structures, and empowers people across the world to easily and securely manage their organizations. In Aragon, tokens represent your stake in the organization, you can utilize crowdfunding to raise funds globally and use voting for more effective results, you can also add a new employee to your organization.

3) *E-Government:* Smart contracts can simplify bureau- cratic processes and improve the efficiency and authority of E-government. For example, Chancheng District in Foshan, China, established the first E-government ser- vice platform using blockchain and smart contracts technology for the sake of improving the quality of government services, developing the individual credit system, strengthening the government's credibility, and promoting the integration of

resources. Other appli- cation areas of smart contracts in E-Government include novel payment systems for work and pensions, strength- ening international aid systems, E-Voting etc.

C. Internet of Things

IoT is an ecosystem of connected physical devices, vehi- cles, home appliances, and other items that are accessible through the Internet. IoT is believed to be widely used in smart grid, smart home, intelligent transportation system, intelligent manufacturing, and other fields. The traditional centralized Internet system is difficult to meet IoT's development needs,

[15]Propy. https://propy.com/.

[16]Ujo Music. https://ujomusic.com/.

[17]Aragon. https://aragon.org/.

Such as the security of sensitive information and trusted interaction between multidevices. Therefore, the combination of IoT and blockchain becomes an inevitable tendency, and smart contracts will help to automate the complex workflow, promote resource sharing, save costs, and ensure safety and efficiency. We proposed a smart home model based on blockchain and smart contracts, they dis- cussed various interaction processes in the model and proved that the proposed model can significantly reduce the daily management costs of IoT devices through simulation experiments. We proposed a smart contract-based framework, which consists of multiple access control con- tracts, one judge contract, and one register contract, to achieve distributed and trustworthy access control for IoT systems. Iotex[18] is a privacy-focused blockchain-driven decentralized IoT network that supports multiple IoT ecosystems, including shared economy, smart home, identity management, and supply chain.

D. Energy

With the rise of energy revolution, the future development trend of the energy industry is distributed and clean energy. Blockchain technology can be used to build distributed energy system and deploy energy supply and trading smart contracts, so as to build the decentralized energy trading markets, improve energy utilization efficiency, and reduce grid operating costs. At present, the main application scenarios of energy blockchain projects include distributed energy, electric vehicle, energy trading, carbon tracking, and registries. Exergy[19] is a consortium blockchain platform that creates localized energy marketplaces for transacting energy across existing grid infrastructures. On the Exergy platform, prosumers who generate the energy through their own renewable resource can transact energy autonomously with consumers in their local marketplace. The Sun Exchange[20] is a blockchain- enabled marketplace that enables its members to purchase and then lease solar cells to schools, businesses, and com- munities in the sunniest locations on Earth (mainly Africa) and its members' earnings are calculated on the amount of electricity their solar cells have produced. The Sun Exchange will arrange the monthly lease rental collection and distribu- tion. Knirsch *et al.* [50] presented a reliable, automated, and privacy-preserving selection of charging stations based on pric- ing and the distance to the electric vehicles. The proposed protocol was built on a blockchain where electric vehicles signal their demand and charging stations send bids.

There are some other application scenarios of smart con- tracts, e.g., healthcare, prediction markets, intelligent transportation system, etc.

FUTURE DEVELOPMENT TRENDS

In this section, we will introduce the future development trends of smart contracts from three aspects, namely, formal

[18]Iotex. https://iotex.io/.

[19]Exergy. https://exergy.energy/.

[20]The Sun Exchange. https://thesunexchange.com/.

verification, Layer 2, and smart contracts-driven parallel orga- nizational/societal management.

A. Formal Verification

Formal verification means applying a proof that the pro- gram behaves according to a specification. In general, this is done with a concrete specification language used to describe how input and output of functions are related. Formal ver- ification of smart contracts involves proving that a contract program satisfies a formal specification of its behavior. It corresponds to the operations layer in the research frame- work we proposed. Hirai defined a formal model for the EVM using the Lem language. The proposed model proved safety properties of a smart contract using the interactive the- orem provers. We extended an existing EVM formalization in Isabelle/HOL by a sound program logic at the level of bytecode. Hildenbrandt presented KEVM, an executable formal specification of the EVM byte- code stack-based language built with the *K* Framework, which designed to serve as a solid foundation for further for- mal analyses. Bhargavan outlined a framework to analyze and formally verify the functional correctness and run- time safety of Ethereum smart contract by translating both Solidity program and EVM bytecode into F*, a functional programming language aimed at program verification. Most of these formal verification tools are still in the experimental stage and have not been widely used. In the future, formal verification will become an important research direction as it provides the highest level of confidence about the correct behavior of smart contracts.

B. Layer 2

As mentioned earlier, smart contract faces many challenges, e.g., poor performance, inability to handle complex logic exe- cution and high-throughput data, lack of privacy protection, and inability to implement cross-chain. A viable solution in the future is called Layer 2. It corresponds to the infrastruc- tures layer in our research framework. Layer 2 creates an off-chain contract execution environment, where blockchain acts as the "consensus layer" which is responsible for the tran- sition of contract-related states and token payment, thereby separating the execution of smart contracts from the consen- sus process of the blockchain and thus realizing high level of performance and privacy. One implementation of Layer 2 is Off-Chain State Channels, which provide state maintenance services between different entities by establishing a bidirec- tional channel between different users or between users and services. State Channels allow performing transactions and other state updates off-chain, while still having full confi- dence that they can revert back to the main-chain if necessary. In addition, Plasma [58] can conduct off-chain transactions by allowing for the creation of "child" Ethereum blockchains attached to the main-chain while relying on the underlying blockchain to ground its security. Truebit[21] makes it possible to perform computationally expensive computations off-chain as a smart contract.

[21]Truebit. https://truebit.io/.

C. Smart Contracts-Driven Parallel Organizational/Societal Management

The rapid development of the Internet and its deep coupling with the physical world have fundamentally changed the management pattern of modern organizations and societies. The future development trend of organizations/societies is bound to a transformation from cyber-physical systems to cyber-physical-social systems (CPSSs) in which social and individual factors must be taken into account . At present, the concept of parallel societies based on CPSS has sprouted, and their substantive characteristics are uncertainty, diversity, and complexity due to the social complexity.

Blockchain and smart contracts are the infrastructures for implementing the CPSS-based parallel organizations/societies because they provide effective decentralized data structures and interaction mechanism for distributed social systems and distributed AI. As mentioned earlier, nodes running smart contracts can be regarded as software agents who have an understanding of the external environment and act upon it. Since different nodes represent the interests of different individuals in an organization/society, they deploy and exe- cute contracts through autonomous negotiation, thus forming various DAOs/DACs/DASs. Beyond the traditional organiza- tions/societies that organized in a hierarchical structure and top-down commands, DAOs/DACs/DASs can help to solve the main problem in organizational management domain, namely, principal-agent dilemma.

The artificial societies computational experiments parallel execution (ACP approach, where artificial systems are used for modeling and representation, computational exper- iments are utilized for analysis and evaluation, and parallel executions are conducted for control and management of complex systems) is by far the only systematic research framework in the field of parallel organizational/societal management. Wang proposed the conceptual framework, fundamental theory, and research methodology of parallel blockchain. We believe that the ACP approach can be naturally combined with blockchain

to realize smart contracts-driven parallel organizational/societal management. First, the P2P network, distributed consensus, and incentive mechanism of the blockchain are the nature ways of modeling a distributed system, each node will act as an autonomous agent and eventually constitute software-defined organization/societal systems (corresponding to artificial soci- eties). Second, the programmable feature of smart contract enables a variety of WHAT-IF-type virtual experimental design, experimental scenarios deduction, and experimental results evaluation (corresponding to computational experiments), so that the agents can make the optimal decision in a specific scenario. Finally, the combination of blockchain and IoT can generate a wide variety of smart assets, making it possible to connect physical world and virtual cyberspace. Through the virtual-real interactions and parallel evolution between the physical and artificial organizations/societies, the optimal organizational/societal management scheme can be obtained (corresponding to parallel execution). This corre- sponds to the manifestation layer in the research framework we proposed.

CONCLUSION

With the increasing popularization and deepened applica- tions of blockchain technology, emerging smart contracts have become a hot research topic in both academic and industrial communities. The decentralization, enforceability, and verifia- bility characteristics of smart contracts enable contract terms to be executed between untrusted parties without the involvement of a trusted authority or a central server. Thus, smart contracts are expected to revolutionize many traditional industries, such as financial, management, IoT, etc. In this paper, we present a comprehensive overview of smart contracts, including the operational mechanism, mainstream platforms, and application scenarios. Specially, we propose a basic research framework of smart contracts based on a novel six-layer architecture. Then we discuss the open challenges standing ahead of smart con- tracts and the recent research progresses. Finally, the future development trends are discussed. The focus of this paper is to make a systematic review of smart contracts and identify some research gaps that need to be addressed in future studies.

REFERENCES

[1] N. Szabo. (1996). *Smart Contracts: Building Blocks for Digital Markets*. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/ Information-InSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html

[2] N. Szabo. (1997). *The Idea of Smart Contracts*. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/ Literature/LOTwinterschool2006/szabo.best.vwh.net/idea.html

[3] J. Stark. (2016). *Making Sense of Blockchain Smart Contracts*. [Online].

Available: https://www.coindesk.com/making-sense-smart-contracts/

[4] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[5] Y. Yuan and F.-Y. Wang, "Blockchain and cryptocurrencies: Model, techniques, and applications," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 9, pp. 1421–1428, Sep. 2018.

[6] X. Xu *et al.*, "The blockchain as a software connector," in *Proc. 13th Working IEEE/IFIP Conf. Softw. Archit. (WICSA)*, 2016, pp. 182–191.

[7] *Ethereum Yellow Paper*. (2018). [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[8] K. Bhargavan *et al.*, "Formal verification of smart contracts: Short paper," in *Proc. ACM Workshop Program. Lang. Anal. Security (PLAS)*, Vienna, Austria, Oct. 2016, pp. 91–96,

[9] M. Risius and K. Spohrer, "A blockchain research framework: What we (don't) know, where we go from here, and how we will get there," *Bus. Inf. Syst. Eng.*, vol. 59, no. 6, pp. 385–409, 2017.

[10] X. Xu *et al.*, "A taxonomy of blockchain-based systems for architec- ture design," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, 2017, pp. 243–252.

[11] F. Glaser, "Pervasive decentralisation of digital infrastructures: A frame- work for blockchain enabled system and use case analysis," in *Proc. 50th Hawaii Int. Conf. Syst. Sci.*, 2017, pp. 1543–1552.

[12] L. Luu, D. H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, Vienna, Austria, Oct. 2016, pp. 254–269.

[13] D. S. Modha *et al.*, "Cognitive computing," *Commun. ACM*, vol. 54, no. 8, pp. 62–71, 2011.

[14] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, May 1996.

[15] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge, "The belief-desire-intention model of agency," in *Proc. Int. Workshop Agent Theories Archit. Lang.*, 1998, pp. 1–10.

[16] *What is a DAO?* Accessed: Oct. 17, 2018. [Online]. Available: https://blockchainhub.net/dao-decentralized-autonomous-organization/

[17] L. Lotti, "Contemporary art, capitalization and the blockchain: On the autonomy and automation of art's value," *Finance Soc.*, vol. 2, no. 2, pp. 96–110, 2016.

[18] A. Dika, "Ethereum smart contracts: Security vulnerabilities and security tools," M.S. thesis, Dept. Comput. Sci., Norwegian Univ. Sci. Technol., Trondheim, Norway, 2017.

[19] Y. Yuan and F.-Y. Wang, "Towards blockchain-based intelligent trans- portation systems," in *Proc. IEEE 19th Int. Conf. Intell. Trans. Syst. (ITSC)*, Rio de Janeiro, Brazil, 2016, pp. 2663–2668.