IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 14, 2019, accepted April 3, 2019, date of publication April 12, 2019, date of current version April 26, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2911031

Security, Performance, and Applications of Smart Contracts: A Systematic Survey

SARA ROUHANI AND RALPH DETERS
Department of Computer Science, University of Saskatchewan, Saskatoon, SK S7N5C9, Canada
Corresponding author: Sara Rouhani (sara.rouhani@usask.ca)

ABSTRACT Blockchain is the promising technology of recent years, which has attracted remarkable attention in both academic studies and practical industrial applications. The smart contract is a programmable transaction that can perform a sophisticated task, execute automatically, and store on the blockchain. The smart contract is the key component of the blockchain, which has made blockchain a technology beyond the scope of the cryptocurrencies and applicable for a variety of applications such as healthcare, IoT, supply chain, digital identity, business process management, and more. Although in recent years the progress toward improving blockchain technology with the focus on the smart contract has been impressive, there is a lack of reviewing the smart contract topic. This paper systematically reviews the key concepts and proposes the direction of recent studies and developments regarding the smart contract. The research studies are presented in three main categories: 1) security methods and tools; 2) performance improvement approaches; and 3) decentralized applications based on smart contracts.

INDEX TERMS Smart contract, blockchain, review, security, performance, application.

I. INTRODUCTION

Blockchain is a trustable distributed ledger that replicates and shares data between peer to peer network. Blockchain initially introduced by an anonymous author, Satoshi Nakamoto, who developed bitcoin to transfer digital currencies directly without the need of third parties [1]. As the name implies, blockchain is a chain of chronological blocks. Each block is identifying by its hash value and links to the previous block by referencing the hash of the previous block, as you can see in figure 1. The only exception is the first block (called ''Genesis block''), which does not include the hash value of the previous block and can be considered as the ancestor block.

Blockchain applications are not limited to monetary transactions. It is possible to program complex transactions, called smart contract, which runs automatically. Smart Contract is a program that is stored on blockchain like other transactions and automatically enforces its terms without the help of trusted intermediaries. This has made a significant mutation in blockchain world and has created a new era for blockchain. Utilizing smart contracts allows us to employ blockchain in several fields beyond cryptocurrency applications such as healthcare [2]–[13], supply chain [14]–[19], business process management [20]–[25], Internet of Things [26]–[32], Digital identity [33]–[38], Record Keeping [39]–[43], Voting [44]–[46] and more [47]–[50]. A review of major applications based on smart contract is presented in section V.

Although smart contracts have been utilized recently, the idea discovered a long time ago. About twenty years ago, the idea of smart contract expressed by a cryptographer researcher, Nick Szabo [51], [52]. The base idea is embedding contractual concepts such as liens, bonding, etc. in the computer components. He introduced four basic objectives of contract design: observability, verifiability, privity, and enforceability. Based on Szabo idea, contract parties are able to observe their performance, verify that the contract has been performed or breached. Also, smart contract preserves the privacy of both parties and distribute the information as much as is needed, and finally it should be run self-enforce. However, the required technology and protocol was not available in that time and the idea remained just as a theoretical concept, today by advancing the blockchain technology smart contract implementation has become realistic.

There are several generic reviews of blockchain technology such as [53], [54] or review papers that target a specific aspect such as security [55] or decentralized

The associate editor coordinating the review of this manuscript and approving it for publication was Kaigui Bian.

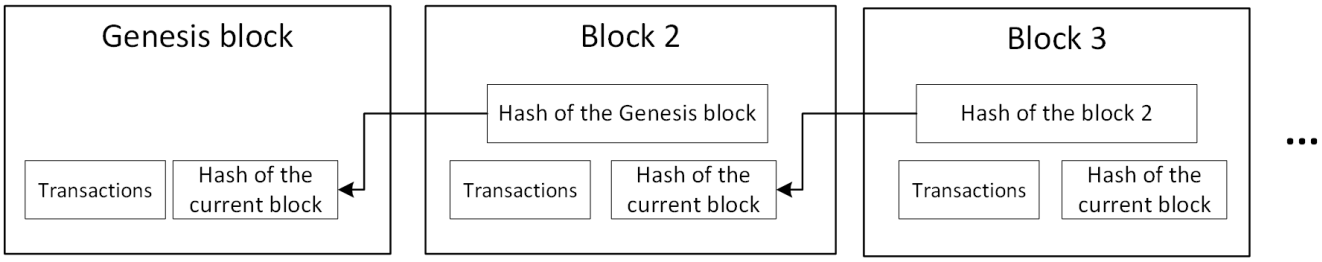

FIGURE 1. Blocks connection in blockchain.

TABLE 1. Summary of the selected papers.

Scientific database	Total number of papers	Journal	Conference	Symposium	Workshop	Preprint	Other
IEEE	31	4	23	2	2	-	-
ACM	17	2	11	1	3	-	-
Springer	9	-	7	1	1	-	-
Elsevier	2	2	-	-	-	-	-
MDPI	3	3	-	-	-	-	-
Wiley	2	2	-	-	-	-	-
arXiv (preprint)	16	-	-	-	-	16	-
IACR Cryptology Archive	1	-	-	-	-	-	1
Other Journals	4	4	-	-	-	-	-
Other Conferences	4	-	4	-	-	-	-
Other Symposiums	1	-	-	1	-	-	-
Total	90	17	45	5	6	16	1

application [56], [57], or particular application such as healthcare [58], [59] or IoT [60]. However, none of them studied smart contract topic particularly. Although, there is another review regarding smart contract [55], but it is limited to Ethereum platform and security aspect. To our knowledge, our paper is the first review of smart contract topic, which investigates extensive aspects, considerations, limitations, platforms, and applications in this field.

We followed Kitchenham *et al.* [61] and Petersen *et al.* [62] systematic mapping study methods to provide an extensive review of smart contract in distributed ledger technology. We defined following questions regarding smart contract topic in blockchain technology.

Q1: What are the available platforms and programming languages for smart contracts?

Q2: What are the limitations of smart contracts?

Q3. What are the presented decentralized applications, which particularly discuss smart contracts design and implementation?

Q4. What are the future directions and research gaps?

To address these questions, we identified relevant papers related to smart contract using a keyword search include "smart contract", "contract" , "chaincode" (chaincode is the term used by Hyperledger Fabric platform [63] for smart contract) in scientific databases including but not limited to IEEE, ACM, Springer, Elsevier, MDPI, Wiley and preprint arXiv. The studies that only mentioned smart contract concept, but they did not address the smart contract design features or concerns in the context of represented methods or application are excluded from our study. Among all identified papers we selected 90 research studies from different scientific databases include 16 journal papers, 45 conference papers, and the rest are workshop, symposium, and preprint papers. After we meticulously investigated papers from arXiv preprint database, we decided to selected 16 papers from arXive preprint database based on the relevance and the quality of the paper. Table 1 shows the selected papers statics based on different scientific databases.

After reviewing and categorizing the selected papers to address the second pre-mentioned question that asks about the smart contract challenges, we recognized two main problems with smart contract design, implementation, and execution. First, the security of the smart contract, and second the performance of smart contract execution. Therefore, we decided to assign separate sections to investigate the security and performance of smart contracts completely. As a result, the following sub-questions is also answered in this paper.

q1: What are the security problems in smart contracts and how we can address them?

q2: What methods are applied to improve the performance of smart contracts?

This paper presents the following contributions:

- It reviews and categorizes smart contract platforms, and domain specific programming languages for smart contract.
- It introduces smart contract challenges and limitation.
- It systematically reviews studies, which recognize a problem in smart contract and provide a novel solution to resolve the problem.

TABLE 2. Blockchain platforms.

Platforms	Consensus	Public or Permissioned	Supports Smart Contract	Smart Contract Language	Built-in Cryptocurrency
Bitcoin	PoW	Public	Yes(with support of side chains)	Ivy[3], RSK[4], BitML [65]	Yes (Bitcoin)
Ethereum	PoW and PoS	Both	Yes	Solidity, Flint[66], SCILLA [67]	Yes (Ether)
Hyperledger Fabric [5]	PBFT	Permissioned	Yes	Go, Node.js, Java	No
Neo [6]	dBFT	Both	Yes	C#, VB.Net, F#, Java, Kotlin, Python	Yes (NEO
Nem [7]	Proof of importance	Both	Yes	Provide template	Yes (XEM)
Quorum [8]	Raft-based and Istanbul BFT	Permissioned	Yes	Solidity	No
Cardano [9]	PoS	Public	Yes	Plutus (Functional Language)	Yes(ADA)
EOS[10]	DPoS	Public	Yes	C++	Yes(EOS)
R3 Corda[11]	Flexible plug-in feature for consensus	Permissioned	Yes	Kotlin	No
Tendermint[12]	BFT	Permissioned	Yes	Any Language	No
Waves[13]	LPoS	Public	Yes	RIDE	Yes(Waves)

- It presents state of the decentralized applications with the focus on smart contract.
- It discusses future directions and research gaps.

The remainder of this paper is structured as follow: Section II reviews key concepts, blockchain platforms that support smart contract, and smart contract programming languages. The different security methods to detect smart contract faults before the deployment and smart contract security analysis tools is explained in section III. Section IV examines the methods that improves the performance of executing smart contracts and smart contract performance analysis tools. Section V surveys blockchain-based decentralized application with focus on smart contracts. Finally, section VI, concludes this survey by presenting the summary of the contributions and challenges, and outlining future directions.

II. PLATFORMS OVERVIEW AND KEY CONCEPTS

A. PUBLIC AND PERMISSIONED BLOCKCHAIN

Public blockchain initially introduced by Bitcoin, which its nodes are untrusted. In addition to public blockchain, there is also permissioned or private blockchain.

Public or permission-less blockchain is open to the world. Everybody with an anonymous identity can join the blockchain, input transactions, and participate in the consensus process. The computational power in public blockchain significantly increases as the number of blocks and the total size of data grows. Currently, most public blockchains use a category of PoW consensus mechanism, which is explained in the next subsection.

Permissioned or private blockchain works similar to the public blockchain, but there is a membership layer on top to authenticate users before joining blockchain, and only permissioned users can join the blockchain and they could have different access levels for submitting the transactions, reading the transactions, or participating in consensus mechanism. Because of the initial user filtering, permissioned blockchain can use lighter consensus mechanism (semi-centralized consensus methods [64]) so, they process transactions faster. Hyperledger Fabric [63] is a well-known example of the permissioned blockchain.

B. PLATFORMS

Bitcoin[1] is the first platform that utilized blockchain and mainly developed for managing and transferring Bitcoin (its cryptocurrency). After Bitcoin, emerging smart contract by Ethereum [2] leads to a new generation of blockchain systems along with various applications. In continue we overview the main blockchain platforms that support smart contract. Table 2 compares these platforms in different feature include consensus method, network type, smart contract language and whether they have their own currency or not.

C. CONSENSUS MECHANISM

Consensus mechanism determines the validating process of blocks, controlling the malicious behavior and reaching consensus between all peers. Since the development of distributed ledger technology several consensus mechanisms have been proposed, which are different in consumption power, performance, scalability, and tolerating malicious behaviors. Wang *et al.* [64] reviews blockchain consensus algorithms with the focus on both distributed consensus system design and incentive mechanism design for permission-less blockchains. Nguyen and Kim also [68] present a survey on consensus algorithms. They categorize the consensus mechanisms into two main groups: proof-based and voting-based. In the proof-based method, one leader (or a group of multiple leaders) is selected and is

[1] https://bitcoin.org/en
[2] https://www.ethereum.org/

responsible for validating and appending a new block to the blockchain. Their main difference is how the leader can be selected. In the voting-based, multiple nodes vote for each block validation and based on consensus policy a minimum positive are required for block validation. In the follow a list of consensus mechanisms, which mainly applied by the blockchain platforms based on table 2 are introduced.

Proof of Work (PoW) [69] is the consensus mechanism proposed by S. Nakamoto for Bitcoin and then applied by Ethereum as well. Proof of work is an incentive-based approach that means nodes (called miner nodes) should solve a complicated mathematical puzzle to earn rewards. The process is like a frequently guessing until the puzzle is solved, and a value called nonce is reached. The first miner who solves the puzzle is the winner of the current block. Then the block broadcasts to other nodes for verification. The verification process is not a heavy task like the initial puzzle. After nodes check the proposed block against frauds, the set of ordered transactions will be committed as a new block to the blockchain. PoW requires a high computation and energy power, however it is very resilient against tampering.

Proof of stake (PoS) initially introduced by Peercoin [3] to reduce the cost of PoW. It is based on the proof of ownership of the cryptocurrency. In each round, the miner is chosen based on node's stake value. As much as the node is wealthier the chance to be chosen is more. The miner gets rewarded by proposing a correct block after validating by other nodes. Pos reduce the amount of computation, but there is a potential problem that rich get richer every time. That means wealthier nodes have a better chance to get selected each time.

Delegated Proof of Stake Delegated Proof of Stake (DPoS) [70] is similar to PoS, but only a subset group of nodes (called witness) can participate in the block production process. The block producer members are selected by stakeholders. Simply any nodes who own any amount of token is considered as a stakeholder. Since the number of validator nodes are less, the process is faster and more efficient.

Proof of Importance (PoI) has been presented by NEM blockchain platform [71]. Each account is assigned a rating based on its importance using graph theory techniques. Accounts with higher importance have higher chance to attach a new block. As much as the account uses blockchain and transfers coins, 0 its important rate increases. Unlike PoW, PoI does not requires high power.

PBFT [72] is based on the Byzantine Fault Tolerate method. First, a leader is selected. The leader is not permanent and will be replaced frequently. Each round includes three phases pre-prepared, prepared, and commit for adding a new block to the blockchain. First, in pre-prepared phase, the leader orders the transactions and proposes a new block as a proposal to the remaining nodes. In prepare phase, the other nodes broadcast their votes to the leader and other nodes. Finally, if only two-thirds of the nodes accept the proposal, a new block will be approved and committed to the blockchain. Figure 2 shows the three phases in PBFT. PBFT can handle less than 33 malicious nodes, so it is suitable for permissioned blockchain because an initial filtering to participate in consensus mechanism is applied. PBFT is not scalable when the number of nodes grows. Hyperledger Fabric uses PBFT consensus mechanism.

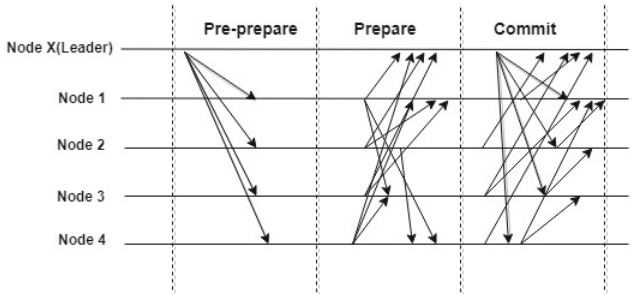

FIGURE 2. PBFT three handshaking phases.

Raft [73] is a fast consensus mechanism used by Quorum and R3 Corda platforms. There are three states in Raft, leader, follower, and candidate. The system starts from follower state. After that specific times out, if a follower does not hear anything from the leader it transforms to candidate state. The nodes vote for the next leader. The candidate that receives maximum votes becomes the next leader, otherwise it remains in the election states or returns to the follower state.

Istanbul BFT (IBFT) [74] is inspired by PBFT [72] with some modification. Similar to PBFT it is three-phase consensus, pre-prepared, prepare, and commit. The system can tolerate F faulty nodes if we have N validators and $N = 3F+1$. IBFT is a final protocol, it means the fork is not possible. In IBFT, there is no client to send the proposed block and every validator can propose the suggesting block. In each round, one validator is selected by other validators and the selected validator broadcasts a new block (pre-prepare message). Then validators that received pre-prepare message broadcast prepare message. If validators receive $2F+1$ prepare message, they enter to prepared phase and then if they accept the proposed block, they broadcast commit message. If validators receive $2F+1$ commit messages and they enter to commit phase the block will appended to the blockchain. Istanbul BFT has been employed by Quorom blockchain.

D. SMART CONTRACTS PROGRAMMING LANGUAGE

Some blockchain platforms such as Hyperledger Fabric, Neo, EoS, and Tendermint support general programming languages such as Java, C++, NodeJS, Python, and Go for smart contracts, however, some other platforms such as Ethereum presented particularly languages for writing smart contracts such as Solidity. In this section, we overview the programming languages that particularly designed for developing smart contracts. Table 2 mentioned the supporting platforms for these smart contract languages. In general there are two main trends for developing new programming language for smart contracts. First, using logic and specifics to make smart contracts easier to understand and more similar to traditional

[3]https://peercoin.net/

TABLE 3. Smart contracts programming language.

Blockchain platform	Programming language for creating script	Turing complete	Computation level	Virtualization technology
Bitcoin	Script	No	Low-level (Virtual Machine)	No
Ethereum	EVM bytecode	Yes	Both Low-level and High-level (Ethereum Virtual Machine) (Solidity and serpent)	No
Nxt	There is no scripting functionality in Nxt	No	High-level (JavaScript)	No
Hyperledger fabric	Script	Yes	Chaincode (Golang, node.js, Java)	Yes (Docker)

contracts. The second studies aim to develop a programming language to improve the security of the smart contracts.

Seijas *et al.* also review the scripting languages used in three blockchain systems [75] in particular Bitcoin, Ethereum and Nxt. Table 3 shows the summary and the specification of represented smart contract scripting languages plus Hyperledger fabric that we added for the completion.

1) SOLIDITY

Solidity [76] is the most popular programming language particularly designed for writing smart contract. Solidity is an object-oriented, turing complete language and runs on Ethereum Virtual Machine (EVM). Ethereum is not the only blockchain platform that supports Solidity, Quorum is also uses Solidity and EVM. Even platforms such as Hyperldger Fabric that designed to support general programming language (Node, Java, and go) for writing smart contracts has integrated solidity smart contracts as well [77]. Solidity is the most well-known and mature contract-oriented language, however it is vulnerable against security attacks [78], therefore many researchers have motivated to improve the security of the Solidity smart contracts by proposing a mid-level language or smart contract analyzing tools.

2) LOGIC-BASED SMART CONTRACTS

Idelberger *et al.* suggest logic-based or declarative language as a replacement of procedural languages [79] to make smart contracts more similar to traditional contracts. The advantages of logic-based smart contracts are as follow:

- They are more understandable for contracts parties.
- The contracts parties negotiations can be transferred to logic-based contract easier.
- They can be validated easier.
- In terms of storage, logic-based contracts are lighter.
- They can ease the interoperability between contracts by applying rule interchange languages.
- The process of contract modification is easier.

First, the paper examines a Pseudo-code of the licensing contractual clauses as an example of a typical contract and then represents the Formal Contract Logic (FCL) implemented by defeasible logic engine SPINdle [80]. For the implementation of logic-based smart contracts, the study suggests two possible solutions include off-chain and on-chain. In off-chain option, there is a centralized system at top of the blockchain and a centralized server execute the smart contract. In on-chain implementation, the smart contract can

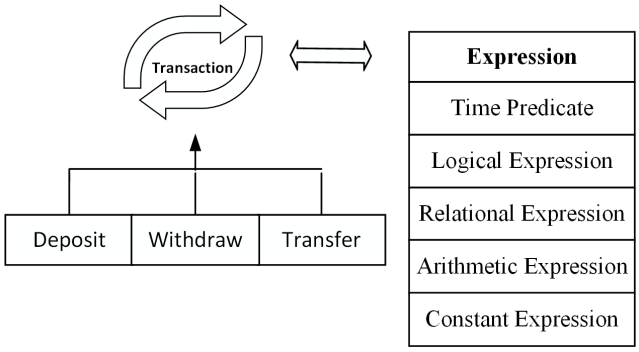

FIGURE 3. SPESC supported expressions and transactions.

form and negotiate off-chain or on-chain, the following steps include contract storage, enforcement and monitoring, and the modification needs to be executed on-chain and stores on blockchain. The noticeable challenges recognized as a result of the implementation of logic-based smart contracts, so it requires further studies to implement more efficient and cheaper logic-based algorithm.

3) SPESC (A SPECIFICATION LANGUAGE FOR SMART CONTRACTS)

SPESC is a specification language for smart contract and as the name implies, it determines the specifications of a smart contract [81]. It creates an abstraction contract on the top of the smart contract, which written by another programming language. SPESC has been designed to make smart contracts understandable for different collaboratives who are involved in the process of smart contract design such as business experts and lawyers like logic-based smart contracts [79]. It presents smart contracts in a similar way as real-world contracts by the specification that describes each party, a set of contracts terms, commitment and rights of the parties, and the conditions of the contract. SPESC smart contract includes four components, contract parties, properties, terms (include obligations and rights as subclasses), and data type definitions (include the primitive type and complex type as subclasses). SPESC supports various expressions to define term conditions. Figure 3 shows SPESC supported expressions and transactions. The paper also examines the understandability of SPESC by running an experiment based on questionnaires over fifteen participants from computer science and law departments. The results demonstrate that SPESC is more understandable in comparison with solidity smart contracts, however, the paper did not translate SPESC

to solidity or other smart contract programs and deploy it on a real blockchain platform to investigate possible challenges.

4) BAMBOO

To solve the problem of reentrancy (explains in section III) in Solidity smart contracts, Yoichi Harai introduces a programming language to create polymorphic contracts for Ethereum called Bamboo [82]. It forces the state-machine method to programmers. Babmoo's syntax influenced by Erlang. The following code shows three contracts written in bamboo that they deploy together. The contract A can becomes C or abort and contract C can become contract D and their orders is forced by the program.

```
Contract A() {
        case (bool A'){
        // Do something
        Return (true) then become A();
}
case (bool A''()){
        if(something)
                return (true) then become
                C();
        else
                abort;
        }
}
Contract C(){
        case(void C'()){
        if (something)
                //Do something
        else
                return then become D();
        }
 }
Contract D(){
        //Do something
}
```

5) SCILLA (INTERMEDIATE-LEVEL LANGUAGE)

SCILLA [67] is an intermediate-level language for smart contract that applies formal methods to analyze and verify the smart contract written in higher-level languages such as Solidity. They embedded Scilla in Coq (A formal proof management system) to examine the semantics, safety and consistency properties of smart contract. SCILLA aims to achieve expressivity and traceability by representing smart contract in communicating automata and separating computation and communication, effectual and pure computation as well as separating invocation and continuation.

6) FLINT

Flint [66] is a domain-specific statically-typed programming language for Ethereum smart contract, which addresses security. Flint includes caller capabilities blocks to check the access permission of Ethereum accounts and contract's functions. Creation, duplication, and destruction of assets are not allowed in Flint, but they can be split, merged or transferred. Flint also categorizes smart contracts functions into two groups, mutating function, which changes the contract state and the other is non-mutating functions. In flint calling the mutating functions by non-mutating functions is restricted and is not possible.

7) BITML (BITCOIN MODELLING LANGUAGE)

BitML [65] is a high-level programming language for Bitcoin smart contracts to define the terms of exchanging Bitcoins. It creates smart contracts in the form of symbolic expressions (symbolic model), then compiles these expressions to Bitcoin scripts. Symbolic expressions can be easier analyzed using formal methods and they argue that any violation at the computation level is recognizable at the symbolic level as well. BitML can implement many of the Bitcoin smart contract services such as escrow services, timed commitments, lotteries, gambling games, but it does not support express contingent payments (contracts that allow selling solutions for a class of NP problems).

III. SMART CONTRACTS SECURITY

Smart contracts could handle a large sum of money, digital assets, stoke, or data. Considering the security aspect of smart contracts is very important since even a tiny bug can lead to significant problems like lots of money lose or privacy leakage. For example, Ethereum well-known crowdfunding smart contract, DAO (Decentralized Autonomous Organization), was attacked on June 2016 because of the bug in its code and resulted in 60 million USD loss [78]. The attacker exploited the reentrancy vulnerability. The attacker recursively called the split DAO function to transfer Ether (Ethereum cryptocurrency) to her or his owned account and the calls stopped before updating the new balance of the calling contract (the attacker account). Writing secure and bug-free smart contract is a difficult task [83] as previous studies show that a significant percentage of smart contacts, which already deployed on Ethereum blockchain are vulnerable. Luu *et al.* investigate 19,366 Ethereum Smart contracts using their represented tool called OYENTE, and their report shows that more than 45 percent of them are buggy [84].

This section reviews represented methods and tools to address the security issues in smart contract. Table 4 shows a summary of represented methods, which address the security issues in smart contract.

A. CLASSIFICATION OF SECURITY PROBLEMS

Luu *et al.* introduce classes of security problems in Ethereum smart contracts and they propose methods to refine the operational semantic of Ethereum to increase the security of smart contract [84]. They also present OYENTE as a symbolic execution tool, which can detect related bugs in Ethereum smart contract. The smart contracts vulnerabilities are as follow:

1) TRANSACTION ORDERING DEPENDENCE (TOD)

In Ethereum blockchain the order of transactions execution is up to miners and clients do not have any control over them. This problem occurs when there is more than one transaction that invoked by the same contract, and the order of those transactions can affect the new state of the blockchain.

TABLE 4. Smart contracts security methods.

Represented Methods	References
Formal verification	F* [85][86], Isabelle/HOL [87], KEVM [88][89], solidity* & EVM* [85]
Smart contract code analyzer tools	OYENTE [84], EthIR [90], GasTap [91] SmartInspect [92], SECURIFY [93], MAIAN [94], Vandal [95], Smartcheck [96], GASPER [97]
Effective callback	[98]
Using control flow	[99]
New consensus method	[100]
Securely connect blockchain to off-chain	Town Crier [101], Smart Cast [102]
Privacy	Hawk [103], Enigma [104]
Secure Programming Language	Bamboo [82], SCILLA [67], Flint [66]

[4]https://docs.ivy-lang.org/bitcoin/
[5]https://www.rsk.co/
[6]https://www.hyperledger.org/projects/fabric
[7]https://neo.org
[8]https://nem.io/
[9]https://www.jpmorgan.com/global/Quorum
[10]https://cardanodocs.com/introduction/
[11]https://eos.io/
[12]https://www.r3.com/corda-platform/
[13]https://tendermint.com/
[14]https://wavesplatform.com/

The authors suggest guard condition as a solution. The idea is the transaction confirmation becomes dependent on the guard condition satisfaction, otherwise the transaction is dropped.

2) TIMESTAMP DEPENDENCE

This problem is related to the smart contracts, which include conditions that trigger by block timestamp. Block timestamps are set by miners based on their local system time and so they can be manipulated by an adversary. The authors suggest using block index instead of block timestamp because it is incremental and protected them from manipulation.

3) MISHANDLED EXCEPTIONS

This problem targets the contract that call another contract. If any exception occurs in called contract, it terminates and returns false, but it may not notify the caller contract. The paper suggestion for this problem is adding an explicit throw and catch EVM instructions.

4) REENTRANCY VULNERABILITY

This problem backs to DAO vulnerability (explained at the beginning of the section 3). When a contract calls another contract, the current contract execution waits until the called contract finishes. This provides an opportunity for the adversary to exploit the intermediary state of the caller contract and call its methods several times.

B. SMART CONTRACTS SECURITY ANALYSIS TOOLS

1) OYENTE

OYENTE[15] is a tool to analyze Ethereum smart contracts code based on symbolic execution [84]. OYENTE takes two inputs, Ethereum smart contract bytecode and Ethereum

[15]https://github.com/melonproject/oyente

TABLE 5. SmartInspect evaluation result.

Characteristic	SmartInspect	Getter	Decoder
Interactiveness	Yes	Partial	Partial
Distribution	Yes	No	No
Security	Yes	Yes	Yes
Instrumentation	No	No	No
Privacy	Yes	No	Yes
Pluggability	Yes	No	Yes
Consistency	Yes	Yes	Yes
Reusability	Yes	No	No
Unrestricted Types	Yes	No	Yes

global state. It checks contract against four previously mentioned problems in section A. There are four main components include CGFBuilder, Explorer, CoreAnalysis, and Validator. CGFBuilder is responsible for creating a Control Flow Graph of the contract. Explorer symbolically executes contracts. The output of Explorer sends to CoreAnalysis to check the existence of four main problems. Validator removes false positives to make sure OYENTE reports accurate problems to the user. OYENTE's report states that more than 45 percent of the Ethereum contracts include at least one of the four indicated problems based on analyzing 19,366 contracts.

2) EthIR

EthIR is an extension of OYENTE. EthIR acts as a decompiler that modified CFGs provided by OYENTE to create a rule-based representation (RBR) of the bytecodes that is ideal for high-level analyses [90].

3) SMARTINSPECT

Once the contract deployed, it is difficult to inspect the given contract attributes because of the encoded nature of smart contract data. Bragagnolo *et al.* address verifying deployed smart contracts [92]. SmartInspect is a tool to analyze deployed smart contract using decompilation techniques and mirror-based [105] reflection for remotely deployed on a reflection-less system [106] without redeploying the contact (getter method) or using the API to accumulate raw data (ad-hoc decoding method). SmartInspect first parses contract codes to generate AST (Abstract Syntax Tree). Then by interpreting the structured representation of AST, it creates the mirror. In the next step, it uses the mirror to extract contract data then data present into four different formats 1) REST 2) Pharo widget user interface 3) JSON 4) HTML. The authors evaluate the represented tool by comparing SmartInspect with getter and ad-hoc decoder methods in nine different aspects include four characteristics represented in [106] interactiveness, distribution, security, and instrumentations in addition to blockchain important aspects, privacy, pluggability, consistency, reusability, and unrestricted types. Table 5 represents the evaluation results.

4) GasTap

GasTap is a platform that calculates the upper bond (Maximum) for the required amount of gas for Ethereum smart

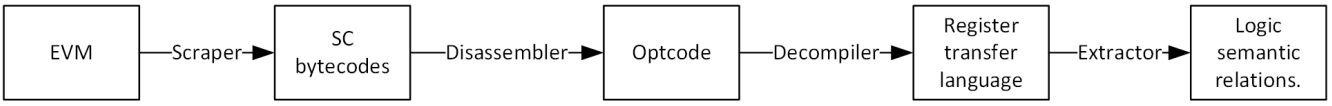

FIGURE 4. Vandal's pipeline.

TABLE 6. SmartCheck smart contract code issue classification.

Security issue	Functional issue	Operational issue	Development issue
Balance equality	Integer division	Byte array	Token API violation
Unchecked external call	Locked money	Costly loop	Compiler version not fixed
DoS by external contract	Unchecked math		private modifier
Send instead of transfer	Timestamp dependence		Redundant fallback function
Reentrancy	Unsafe type inference		Style guide violation
Malicious libraries			Implicit visibility level
Using tx.origin			

contracts to avoid the out of gas vulnerability [91]. It exploits available tools in a pipeline that takes a smart contract and determines the required gas upper limit for its functions. GasTap first uses OYENTE [84] for construction of the CFGs, second it employs an improved version of EthIR [90] for decompiling low-level code to high-level representation, third it uses SACO [107] to determine the size of the relations, forth it generates the gas equations, and finally it utilizes PUBS [108] to solve the gas equations and generate the closed-form gas bands.

5) SECURIFY

Ttsankov *et al.* developed SECURIFY [93], another smart contract security platform. SECURIFY first, decompiles the EVM bytecode of smart contract and extracts semantic facts that are data and control flow dependencies of the contract and finally, it checks the security pattern, which represented in Domain-specific language (DSL). SECURIFY patterns include a set of compliance and violation. All the contract behaviors is labeled by one out of the three labels of Compliance, Violence, and warning. If the semantic fact matches with compliance pattern it is considered as compliance, else if it matches with violence it is considered as violence, else if it matches with neither of the patterns it is considered as warning behavior.

6) MAIAN

MAIAN is an analysis tool represented by Nikolic *et al.* [94] to detect greedy, prodigal, or suicidal behavior of Ethereum smart contracts based on a trace of vulnerabilities. Greedy contract refers to a contract that stays alive, but locks Ether forever. Prodigal contract does not provide any backup solution in case of attacks and they can leak Ether to an arbitrary address. Suicidal Contracts are those, which can be killed by an any arbitrary account by forcing the contracts to commit suicide. Their analysis on 970,898 contracts detected 34,200 (2.365 distinct) vulnerable contracts.

7) VANDAL

Brent *et al.* introduce Vandal as a static security analysis framework for smart contracts [95]. Vandal uses an analysis pipeline to translate smart contract bytecodes to logic relations, which reflecting the program semantics of the smart contract. The analysis pipeline includes a bytecode scraper, a disassembler, a decompiler, and an extractor. Figure 4 shows Vandal pipeline. Also, it exploits Souffle' [109], a declarative language, as a datalog engine to express security vulnerabilities. This study presents a list of five vulnerabilities include unchecked send, reentrancy, unsecured balance, destroyable contract, and use of origin.

8) SmartCheck

Tikhomirov *et al.* classify common Solidity code issues into four main categories, Security, Functional, Operational, and Development [96]. They implemented SmartCheck that translates contract source code (Solidity code) to the XML format and it looks for problematic patterns using XPath queries. Table 6 shows the represented code issues classification.

9) GASPER

Gas is a unit introduced by Ethereum blockchain and it is used for the execution fee that sender of transactions or smart contracts should pay for every operation. In other words, gas is the fee to reward miners for executing the code. Gas can be exchanged with Ether, which is Ethereum cryptocurrency [110]. Chen *et al.* explain a security vulnerability of smart contract that leads to unnecessary gas consumption. They introduce seven under-optimized patterns in smart contracts that classified into two groups: useless-code related, and loop-related [97]. Table 7 shows the seven under-optimized smart contracts patterns based on Solidity programming language. They also developed a tool called GASPER for detecting overcharged three out of seven representative patterns automatically: dead code, opaque predicates, and expensive operations in a loop. The result of scanning 4,240 smart contracts shows that at least 80% of contracts suffer from one of these three patterns and at least 73.2% of smart contracts suffer from two out three patterns and 71.7% of smart contracts involve in the risk of all three patterns.

TABLE 7. Under optimized patterns.

Category	Under-optimized Pattern
Useless code related	1. Dead Code: Existing bytecodes in smart contracts, which will never be executed
	2. Opaque Predicates: The presence of the statement operations in smart contracts, which their results are same.
Loop Related	3. Expensive Operations: Moving expensive operation outside of the loop can lead to saving a significant amount of gas.
	4. Constant Outcome: If the result of the loop's outcome is always constant that loop can be removed.
	5. Loop Fusion: Combining several loops into one loop.
	6. Repeated computations: Sometimes loops can have constant result in each iteration. This pattern saves gas by computing the outcome once and then reusing it.
	7. Comparison with the unilateral outcome: Comparisons that always result in the same value in the loop. They could not recognize in the compilation such as pattern two.

C. FORMAL VERIFICATION

Formal verification is one of the most precise approaches to verify the accuracy of the system and is one of the earliest approaches that is employed to verify the behavior of smart contracts. In this section, we review the studies that address the security vulnerabilities in smart contract using formal methods.

Bhargavan *et al.* propose a framework to analyze and verify smart contracts written in Solidity programming [85]. The authors used F* [86], a functional programming language that works for program verification. They present two tools, Solidity* as a tool to translate Solidity programs to F* programs for verifying the source level functional correctness and EVM*, a tool that works as a decompiler for EVM bytecodes to perform a stack analysis and creates equivalent F* programs for verifying low-level properties. Finally, the tool verifies that both outputs are equivalent. The output of Solidity* checks the dangerous patterns to detect send() function failure exception and reentrancy. The output of EVM* checks the limit of gas consumption of smart contract methods.

Amani *et al.* similarly use the decompilation technique to verify Ethereum smart contracts at the bytecode level by using logical framework Isabelle/HOL [87]. They define the smart contract correctness features by relying on Ethereum termination guarantee gas concept. They split smart contracts bytecode into the basic blocks and create a sound program logic for verification.

Park *et al.* [88] present another formal verification tool to verify EVM bytecodes by adopting KEVM [89], which is a complete executable formal semantics of the EVM. They represent a group of challenges include Byte-Manipulation Operations, Arithmetic Overflow, Gas Limit, and Hash Collision in verifying EVM bytecodes and they propose some techniques to address these challenges. They also instantiate K reachability logic theorem prover [111] to improve the scalability.

D. DETECTION OF EFFECTIVE CALLBACK FREE OBJECTS

Grossman *et al.* present a safety notion called ECF (Effectively Callback Free) and specifically DECF (Dynamically ECF) for executions and SECF (Statically ECF) for objects [98]. Based on the represented definition ''an execution of an object is DECF when there exists an equivalent execution of the contract without callbacks, which starts in the same state and reaches the same final state. An object is considered SECF when all its possible executions are dynamically ECF''. ECF is a bug detection factor and it shows DAO and other buggy contracts are not ECF, in addition, ECF can be used to enable modular reasoning about objects with encapsulated state. An online polynomial time and space algorithm is represented to check the execution of smart contract based on ECF factor and they integrate it into EVM (Ethereum Virtual Machine) [112] with low runtime overhead. This algorithm detects conflict memory accesses and explores commutatively of operations to check the accuracy of conditions. The evaluations result indicates that not all but most non-ECF executions are responsible for vulnerable executions and there are few bug-free non-ECF contracts.

E. CONTROL FLOW

Fröwis and Böhme expose the mutability problem in smart contract control flow in Ethereum blockchain [99]. The paper argues that smart contract control flow is not immutable and has the potential to change. To address this problem, this study initially discusses how the call graph generates from Ethereum smart contracts, then it introduces clean up strategy based on call graph to avoid biases arising from attacks against Ethereum. Eventually, it measures the trustless contracts.

Here are the represented steps to obtain the required content to analyze call relationships in Ethereum smart contracts and create a call graph:

First, they extract Ethereum smart contracts bytecode through JSON_RPC API.[16]

Second, in order to extract the code of the smart contract, they iterate again all transactions and chose those that have no recipient. This technique is limited to contracts, which created by user accounts not those created by code accounts.

Third, they use the feature of tracing mode in parity client[17] to access internal transactions.

Fourth, they store the bytecode, creation block number, and destruction block number of all extracted smart contracts into MongoDB.

[16]https://github.com/ethereum/wiki/wiki/JSON-RPC

[17] https://www.parity.io/

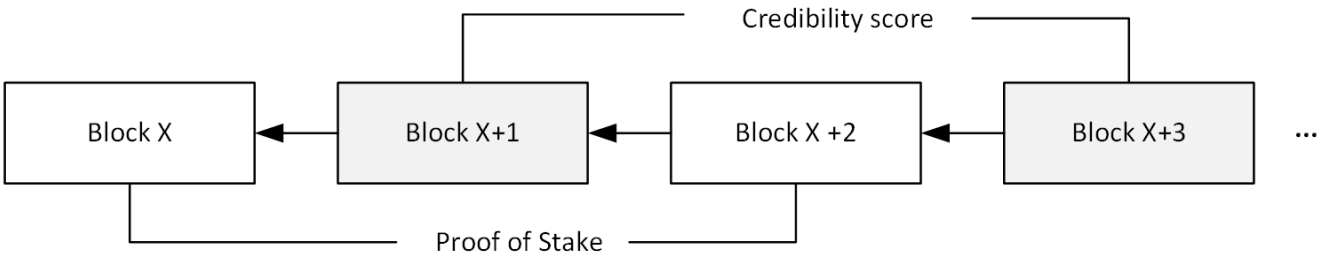

FIGURE 5. Using the hybrid method based on both PoS and credibility score.

Fifth, they use evmdis[18] in order to disassemble the EVM bytecodes and achieve reaching definition to find the source of called addresses.

Finally, the call graph is generated from extracted calls.

The study considers smart contract trustless "if and only if all calls in its dependency tree have hardcoded addresses, hence all code that a smart contract can execute is fixed upon deployment of smart contract". They measured trustless smart contracts before and after cleanup smart contracts affected by popular attacks DAO and DOS (Denial of Service). The results indicate that before cleanup, 54 percent of active contracts were trustless based on the represented definition and after cleanup, the ratio increases to 62 percent, which means two out of five smart contracts deployed on Ethereum require trust in at least one-third party.

F. SECURE CONSENSUS METHOD

Watanabe *et al.* suggest a new consensus method to secure blockchain applied to smart contracts [100]. This study addresses the problem of collapse of coin in proof of stake consensus mechanism. The represented method is based on the collapse of credibility. Credibility is a metric to define trustable contract owners. The credibility of the contract owner is related to how many contracts he or she owns. If a contract owner attacks blockchain, he or she loses trust in whole society. However, using only credibility method increases the chance for fake credibility score and 51% attack. To overcome this problem, using hybrid blockchain based on both proof of stack and credibility score is suggested to address both two problems. Figure 5 shows using the hybrid method in block mining. Based on the hybrid method, the two consecutive blocks should not use the same consensus method.

G. CONNECTING SMART CONTRACT AND OFF-CHAIN

Town Crier is one of the earliest studies that examine connecting smart contracts to an external resource (off-chain) in order to request the concise pieces of data called datagrams [101]. Town Crier is an authenticated data feed system that acts as a bridge between Ethereum smart contract and HTTPS-enabled data source combined with trusted hardware backend.

[18]https://github.com/Arachnid/evmdis/

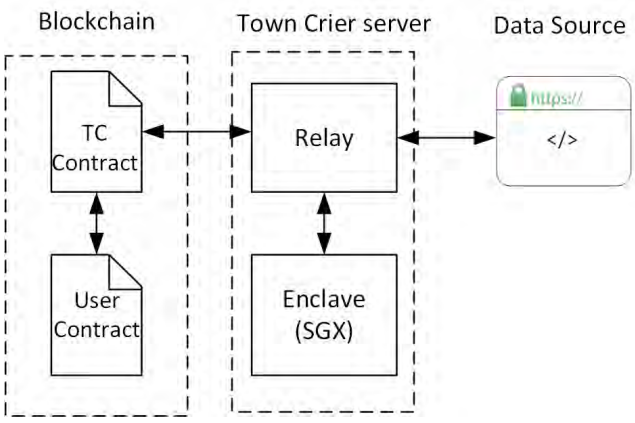

FIGURE 6. Town Crier architecture.

The general trusted and secure architecture model includes three main components: The Town Crier contract, the Enclave, and the Relay. The Town Crier contract acts as a frontend for Town Crier system. It provides an API for the contract that uses Town Crier service. The Enclave is an instance of the Town Crier core program running in the Trusted Execution Environment (TEE), such as SGX enclave and queries data from https data source. The Relay is an ordinary user-space application and it has provided to relay network traffic between smart contract, Enclave, and HTTPS data source. Figure 6 shows Town Crier architecture. The User contract requests a datagram from Town Crier contract. Relay unit relays the request to Enclave and Data Source and finally forward the response to User contract through Town Crier contract.

Kothapalli *et al.* represent SmartCast [102], an incentive compatible consensus protocol that adapted the work of Clement [113] to smart contract. The main idea is using smart contracts to provide incentive compatibility for off-chain consensus protocol. The system financially rewards disinterest parties for consensus and honest participation. The protocol minimizes communication with Ethereum blockchain, so the transaction cost is minimized as well. Since the communication is the major cost, the paper represents this as incentive-compatibility. The essential two components in the smart contract model are: 1) a smart contract program, which

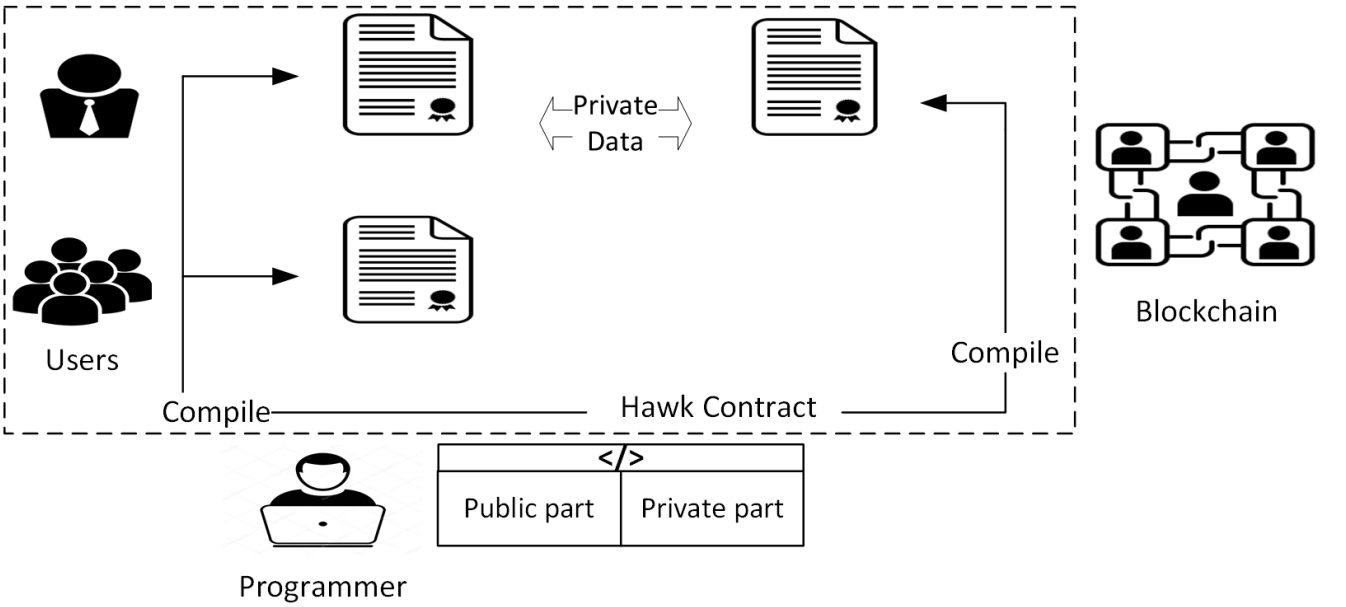

FIGURE 7. Hawk platform architecture.

receives information from nodes about each other's behavior and detects lazy nodes and based on that payout reward to each party. 2) local program for each of the parties to execute, which includes interaction with smart contract and involving in an off-chain consensus process.

Molina-Jimenez *et al.* propose requirements to achieve an effective hybrid method of off-chain and on-blockchain solutions [114]. The hybrid model combines decentralized smart contracts and trusted centralized third parties to address the performance and scalability issues in pure blockchain approaches.

Here are represented practical cases based on the integration of centralized and decentralized approach:

1. Indelible blockchain-based log: the blockchain part can be applied to record passive logs that worth to keep and duplicate in the blockchain.

2. Cryptocurrency-based payment channel: Using public blockchain such as Bitcoin or Ethereum for payment process for the significant amount of money transformation, thus the transaction fee is negligible.

3. Off-blockchain execution of operations: In order to increase the performance of the application and avoid transaction throughput limitation, it is suggested to execute main operations off-chain.

H. PRIVACY

In this section, we overview the studies that address privacy issues in smart contract data or related application.

Hawk is a framework for creating privacy-preserving smart contracts [103]. Hawk receives a plain program (smart contract) and it's compiler automatically generates the cryptographic protocol. A Hawk contract includes two parts, private and public. The private part includes parties' input data and currency units, which are cryptographically invisible from the public views. The rest of the codes, which are not private consider public. Hawk programs are divided into three pieces and they executed by three separated entities. First, the piece that is executed by all consensus nodes, second, the piece that is executed by the users, and third, the piece that is executed by the manager, which is a minimally trusted party and can see users' private data. Hawk guarantees privacy as long as the manager does not disclose the private portion. Figure 7 shows the general overview of Hawk framework. The paper also provides a formal UC-based (Universal Composability) model of cryptography to investigate the security of represented protocol and also can be adopted by other decentralized protocols.

Enigma is a decentralized computation network based on Ethereum that provides privacy along with computation to allow different parties store and share secret data [104]. They have presented a turing-complete scripting language to create private contracts that support private data. The private data are handled through the Enigma off-chain platform and the public parts are executed on the blockchain. The system includes three distributed databases include 1. a blockchain to control the system, manage access control, and as a tamper-proof database of events, 2. a Distributed Hash Table (DHT) for storing off-chain private data, 3. a Multi-Party Computation (MPC) to split data into meaningless chunks and distribute them between different nodes without replication.

Ekiden also uses TEE (Trusted Execution Environment) to achieve privacy and confidentiality for sensitive data related to smart contracts beside high-performance execution [115]. They present a proof of publication method to make sure that

the SGX enclaves have been synchronized with the latest state of the blockchain.

IV. SMART CONTRACTS PERFORMANCE

The performance of executing transactions and smart contracts is a major challenge in blockchain-based systems and this prevents them from competing with current applications in large scale. In this section, we review the studies that aim to improve the performance of smart contracts.

A. CONCURRENT EXECUTION

Dickerson *et al.* present a technique based on transactional boosting methodology [116] to allow miners execute non-conflicting smart contracts in parallel and then capture the parallel execution schedule for validators to achieve deterministic result [117]. In order to avoid problems related to shared data storage, the authors suggest that miners execute contracts code as speculative actions. If data conflicts is detected during the runtime, it can resolve by scheduling and delaying or rolling-back. Miners record the locking schedule of parallel execution in the relative block. This creates a happens-before graph of transactions for validators. Validators convert this schedule into deterministic parallel for-join program to validate the block concurrently. Testing based on three concurrent threads indicates that mining process and validating process can be accelerated up to 1.33x and 1.69x respectively.

To introduce three smart contract execution model, Yu *et al.* suggest breaking down the system states into three hierarchical states: account state, local state, and federal state, while the account system is the smallest state unit [118]. Any change in account state leads to the change in local state and any change in the local state ends up with the change in the federal state. This improves performance by separating the execution of smart contracts from the state management and proposes a pipeline model to verify and create blocks in parallel or concurrent.

1. Sequential execution of smart contracts: It is based on smart contract execution in Ethereum. It is useful for the contracts that their execution time is short, as this model has the longest time for block validation.

2. Parallel execution of smart contracts: If a transaction is recognized as a smart contract, all the nodes execute the smart contract at the same time, so the local state will update concurrently. Nodes verify the local state by comparing it with block state.

3. Non-blocking execution of smart contracts: By separating execution of smart contract from the process of building blocks, accelerate the block building, and validating process. In this model, nodes do not wait for current smart contract to finish the execution and they accept next transactions immediately. In addition, this study proposes decoupling block building from state maintenance in blockchain design and introduce a new blockchain called SBC (State Blockchain) to handle state maintenance including state synchronization and storage.

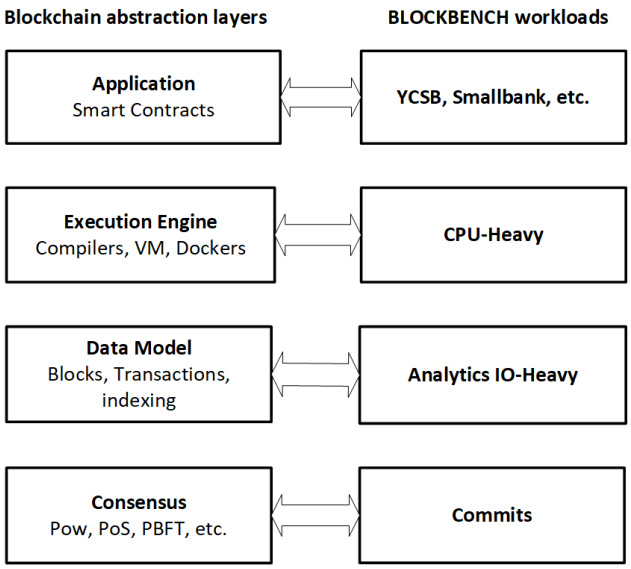

FIGURE 8. BLOCKBENCH abstractions layers and correlated workloads.

Anjana *et al.* [119] propose using Software Transactional Memory (STM) [120] systems to execute non-conflicting transactions of smart contracts concurrently. There are two types of miners in the represented system include serial miner and concurrent miner, both type of miners have access to the same set of transactions. Serial miner executes transactions serially, whereas the concurrent miner executes the transactions concurrently. The concurrent miner must only choose non-conflicting transactions for executing concurrently. If two transactions access the shared data-object and at least one of them performs write operation on it, they have conflict with each other. The concurrent miners identify conflict transaction with the help of optimistic STM system. The validation of transactions can also be executed concurrently. There is a chance that validators re-execute suggested transactions from miners with different order and get a different results, which leads to block rejection. In order to avoid this, the authors suggest concurrent miners provide a conflict graph. Conflict graph includes all the dependencies between conflict transactions in the form of an adjacency list. In summary, every concurrent miner presents a conflict graph along with the set of proposed transactions, the hash of the previous block and the final state of each shared objects.

B. PERFORMANCE ANALYSING FRAMEWORKS

1) BLOCKBENCH

BLOCKBENCH [121] is a framework for performance analyzing of private blockchains. Any private blockchain can be integrated to BLOCKBENCH, however, the paper conducts an evaluation of three blockchain platforms: Ethereum, Hyperledger fabric, and Parity. The study identifies four main abstraction layers for blockchain and then designs four workloads based on them for Blockchain. Figure 8 shows blockchain abstractions layers and correlated workloads in

TABLE 8. BLOCKBENCH implemented smart contracts.

Smart Contract		Description
Macro benchmark workload	YCSB driver[122]	Key-value store
	Smallbank[123]	OLTP workload
	EtherId	Name registrar contract
	Doubler	Ponzi (pyramid) scheme
	WavesPresale	Crowd sale
Micro benchmark Workload	VersionKVStore	Keep state's versions
	IOHeavy	Read and write a lot of data
	CPUHeavy	Sort a large array
	DoNothing	Simple contract, which do nothing

TABLE 9. Performance metrics based on [124].

Overall Performance Metrics	Detailed Performance Metrics
Transaction Per Second	Peer Discovery Rate
Average Response Delay	RPC Response Rate
Transaction Per CPU	Transaction Propagating Rate
Transaction Per Memory Second	Contract Execution Time
Transaction Per Disk I/O	State Updating Time
Transaction Per Network Data	Consensus-cost Time

BLOCKBENCH. For evaluation of represented platforms, the smart contracts related to each workload implemented in two versions of Solidity for Ethereum and Parity and Golang for Hyperledger. Table 8 shows the implemented smart contracts in BLOCKBENCH.

2) REALTIME MONITORING

Zheng *et al.* [124] argue that why Gas is not suitable metric because it is one-sided and inaccurate. They categorize the performance metric of the blockchain systems into two groups: overall performance for the users and detailed performance for the developers. The subcategories of each group are represented in table 9.

They also present a real-time performance monitoring framework to monitor the performance of the blockchain systems, which is scalable and includes lower overhead. Here is the brief explanation of different part of the framework.

- Validating Peer is the essential underlying part of the blockchain to validate and execute transactions and smart contracts. The framework focuses on its performance to calculate the performance of the entire blockchain.
- Log Parser/Analyzer is a terminal corresponding to each validating peer. This part collected the log data about the validating peer and hardware consumption.
- Synchronize Peer helps to collect the rest of the data that is not caught from validating peer without making extra overhead.
- Data collector/calculator collects data from log analyzer and synchronize peer and calculate the performance metrics.
- Web Frontier is the user interface to visualize the collected performance metrics.

Finally, they monitor the performance of 1000 smart contracts on four different blockchain platforms, Ethereum, Parity, Hyperledger Fabric and CITA based on represented performance metrics and the proposed framework metrics. Table 10 shows overall performance on these platforms.

TABLE 10. Performance of different platforms based on real-time monitoring.

Blockchain	Tps	TPC	TPMS	TPDIO	TPND
Ethereum	5.55578	0.00195	0.01060	0.26573	0.22206
Parity-pow	3.95500	0.00140	0.06814	0.00264	0.07167
Fabric	600.611	2.65340	4.28265	0.13816	0.10122
CITA	256.636	1.33393	0.43244	0.59888	0.16208

V. DECENTRALIZED APPLICATIONS

Emerging smart contract has provided the infrastructure for non-financial blockchain-based applications in various domains. Blockchain key characteristics such as decentralized operation, immutable audit trail, data provenance, security, and privacy have made it a suitable alternative for traditional centralized applications and the evolve of smart contract has made it happen. In this section, we review the main application domains that employ blockchain and smart contracts. We categorized them in seven main groups include healthcare, IoT, identity management, record keeping, supply chain, BPM, and voting. However, the blockchain-based applications are not limited to these groups such as [47]–[49]. Tables 11-17 present the studies for each category with emphasis on designing smart contract for the decentralized application.

A. INTERNET OF THINGS

The significant increase in IoT devices in last years leads to exposing insecure big data. The privacy and security of the data produced by IoT devices are the major challenges in this field. Exploiting blockchain technology has investigated by many researchers to address these issues. Samaniego and Deters [125], [126] introduce blockchain as a service for IoT systems. They discuss that blockchain and smart contracts can be applied for the configuration of IoT devices, recording data captures from sensors, and micro-payments. Panarello *et al.* represent a survey to analyze the research related to blockchain and IoT context [60]. They study different application domains and categorize them based on two usage patterns, device manipulation, and data manipulation. they also represent the progress level of the presented applications. In Table 11, the list of related studies in the context of blokchain and IoT with the focus on smart contract is presented. Mainly the represented studies focused on using blockchain and smart contracts to control the access to the data generated by IoT devices as well as addressing the privacy concern [26]–[29].

In addition, a few other studies investigated configuration, registration, resource management, and energy trading in this context as follow.

Huh *et al.* [30] implemented a system based on Ethereum blockchain to configure and synchronize IoT devices by the

TABLE 11. IoT decentralized application.

Reference	Application Feature
Ouaddah et al. [26]	FairAccess is a framework for access control and authorization mechanism based on ethereum smart contracts for IoT systems.
Ouaddah et al. [27]	Pseudonymous and privacy-preserving authorization and access control management framework for IoT systems.
Xu et al. [28]	BlendCAC, federated capability-bases Access control mechanism for IoT systems based on smart contracts.
Cha et al. [29]	Privacy preserved connected gateway for IoT devices. Using smart contract for managing IoT device data and defining privacy policies.
Huh et al. [30]	Using Ethereum smart contract to store the data produces by IoT device, and then configure and define the behavior of the IoT devices.
Ruta et al. [31]	A service-oriented architecture based on semantic blockchain and implementing smart contracts for registration, resource discovery, resource selection, and payment.
Lombardi et al. [32]	Blockchain-based infrastructure to create a reliable and low-cost transactive energy. Energy trade and energy auction are handled by smart contracts.

TABLE 12. Healthcare decentralized application.

Reference	Application feature
Azaria et al. [2]	Patient-centric medical data access control and sharing system (Medrec) based on Ethereum platform. It presents a method to provide anonymous data as an award for miners.
Xia et al [3]	Big Medical data sharing, provenance and audit trail on a cloud platform (Medshare).
Xia et al. [4]	Using permissioned blockchain and integrating identity management service to access to the shared EHR data.
Rouhani et al. [5]	Discretionary access control and mandatory access control based on Hyperledger fabric permissioned blockchain (MediChainTM).
Mikula and Jacobsen [6]	A prototype based on Hyperledger fabric and ChainCodes (smart contract in Hyperledger Fabric), which represents identity and access management in healthcare domain.
Yue et al. [7]	Patient-centric mobile application to share medical data based on Indicator- Centric Schema (ICS).
Peterson et al. [8]	Securely sharing medical data exchange and presenting Proof of Interoperability consensus method by considering FHIR [55]
Theodouli et al. [9]	Proposing a blockchain-based architecture design for sharing medical data.
Nugent et al. [10]	Using smart contract to solve data manipulation issue in clinical trials and proving a trusted immutable record of data.
Shae and Tsai [11]	Presenting a blockchain platform and architecture for clinical trial and precision medicine. Include four main components 1) blockchain-based parallel computing 2) blockchain application data management component, 3) identity management component, 4) trusted data sharing management component.
Zhang et al. [12]	FHIRChain,a blockchain-based architecture design based on ONC (Office of the National Coordinator for Health Information Technology) requirements and FHIR [19] standard.
Zhang et al. [13]	Introducing evaluation metrics to analyze Dapps (Decentralized blockchain-based application) in healthcare domain.

decentralized approach provided by blockchain and manage public key infrastructure (PKI) associated with IoT devices. In their system, the written smart contracts control the configuration of IoT devices such as tracking the value of the meter or defining saving policy values.

Ruta *et al.* [31] propose a framework for Semantic web of Thing (SWoT) based on blockchain and explore designing smart contracts for registration, discovery, and selection of the resources. The registration includes adding a new record to the blockchain and attributes related to the resource. Resource discovery includes reading from blockchain based on filtering metrics defined in the smart contract. Resource discovery selects the best resource based on related smart contract criteria and changes the state of the blockchain.

Lombardi *et al.* [32] present a tamper-proof infrastructure for energy trading within the smart grids. Smart contract implements the energy trading layer to define the policies of trading energy and the security enhancement layer. The energy trading smart contract is responsible for handling energy auctions. The Security enhancement contract detects vulnerable smart meters by information set associated to the smart meter. The vulnerable smart meters are isolated and the transactions related to the vulnerable smart meters get out from the auction.

B. HEALTHCARE

Current healthcare systems suffer from various problems such as scattered data, difficult access, data consistency, interoperability and privacy concerns. In general, most of the studies aim to solve these issues by proposing an architecture design [9], [11] or implementing a system based on blockchain and smart contracts [2]–[8], [10]. The main focus of these studies are the management of users identity management, access control and sharing medical data using different available blockchain platforms such as Ethereum [2], [10], [12] and Hyperledger [5], [6]. Also Kuo *et al.* investigate blockchain key benefits for different healthcare applications such as record management, insurance claim process, clinical research or creating a healthcare data ledger [58]. In this review paper, we chose studies with the focus on smart contracts or at least the smart contract design has been presented in the architecture. Table 12 shows the healthcare applications based on blockchain and smart contracts.

C. SUPPLY CHAIN

Using blockchain data immutability aspect is the main reason for applying blockchain in supply chain applications. Tracking and monitoring the steps from producing products to delivery, ensure quality control, providing an integrable and

TABLE 13. Supply chain decentralized application.

Reference	Application feature
Chen et al. [14]	Exploiting smart contract and blockchain for quality management in supply chain. The real-time quality management, process the quality, and the product quality are monitored by smart contracts and unqualified products automatically are rejected by the smart contract.
Bocek et al. [15]	Using smart contract and IoT sensors to assert data immutability temperature records in pharmaceutical supply-chain and reducing the cost.
Korpela et al., [16]	Investigates using smart contracts for documentation collection trade and automation of transactions, or in general, digital supply chain integration based on DBE framework [128].
Ruta et al. [17]	A semantic-enhanced blockchain platform that supports ontology-based object discovery.
Kim and Laskowski [18]	Analyze a traceability ontology and translate it to a smart contract to achieve supply chain provenance trace and enforce traceability constraints using Ethereum platform.
Figorilli et al. [19]	Tracing electronic records from standing tree to the whole chain of steps using RFID sensors and Azure blockchain workbench and two main smart contracts, timber marking and cutting trees.

the trustable process are the advantages of using distributed ledger technology in the context of supply chain. Table 13 presents the supply chain applications based on blockchain and smart contracts.

Korpela *et al.* [16] investigate digital supply chain integration requirements and consider blockchain technology as an accelerator and cost-effective solution. The data has been collected by interviewing blockchain experts from Finnish business consortium include 30 companies and operated in 36 countries. Based on Quality Function Development (QFD) method [128], the ledger and smart contract are more required blockchain functionalities to support supply chain integration, instead of transactions and hash functionalities. The study explains this by emphasizing on the need for a standardized data model.

Chen *et al.* [14] present a supply chain quality insurance framework, which includes four layers, IoT devices, distributed ledger, smart contract, and business layer. The required data for monitoring the quality of products, assets, and transactions generates by IoT devices. Blockchain layer records the generated data in a secure and trustable ledger. Smart contract handles the privacy by controlling the access to the data and providing a digital identity service. Smart contract layer also enables the intelligence for real-time monitoring, plane logistics data, and predicting the customer requirements. The business layer includes business enterprises and activities.

Similarly, Bocek et al [15] launched a startup called modum.io[20] to monitor and control the quality of the pharmaceutical supply chain. They used IoT devices for measuring the temperature of every parcel containing pharmaceutical products and recorded them on blockchain to make sure about data immutability. Smart contracts are developed to automatically ensure the GDP (Good Distribution Practice of medicinal products) compliance. It means with each shipment a respective smart contract is called and checked if the temperature is accepted automatically. The details of the establishing Ethereum blockchain and Solidity smart contracts are presented in this study.

[20]https://modum.io/

Kim and Laskowski [18] investigate applying ontologies for supply chain provenance. The study explains how to apply TOVE traceability ontologies in blockchain to enhance the blockchain provenance tracking by translating ontologies representation to Solidity smart contracts. Formal ontology axioms represented in first-order logic convert into Ethereum smart contract to generate traces related to intended physical goods. Figurili *et al.* [19] also implemented a blockchain application using Azure blockchain workbench to trace woods from standing trees to final products in order to fight against fraud. The data captured by RFID sensor devices adjust to respective smart contract and record on the blockchain and then become accessible for different parties to trace the history.

D. BUSINESS PROCESS MANAGEMENT (BPM)

The main unsolved issue in the collaborative business process in organizations is the lack of trust, and similar to many other applications using blockchain has been considered as a solution to address this issue without relying on any central authority. Mendling *et al.* [23] investigate blockchain potential in the domain of BPM applications, the challenges for applying this technology and the advantages that it brings for us.

Weber *et al.* [20] implemented three cases for integrating blockchain in BPM by introducing a method to transform the collaborative business process to a factory contract and then run its instances as a smart contract. In continue [21], they represent an optimized resource usage approach to minimize the cost in terms of Gas for executing smart contracts that are responsible for executing business process instances as well as increasing throughput. They translated (Business Process Model and Notation) BPMN to a reduced petri-net model and then compiled petri-net model into Solidity contracts using space-optimized data structure by reducing the total operations and initialization cost in the smart contract.

Pourheidari *et al.* [22] implemented a business process case study of order processing using Hyperledger Fabric permissioned Blockchain and Hyperledger Composer. The details of the implementation include extracting assets from business process, access control component, and configuration.

TABLE 14. BPM (business process management) decentralized application.

Reference	Application feature
Weber et al. [20]	Converting the BPMN model to a factory contract and then implementing a factory contract and running its instances as a smart contract on the blockchain to address the trust issue in the collaborative business process.
García-Bañuelos et al. [21]	Presents an approach to convert a business process model into a Petri-net model and then compile it to a minimized solidity smart contract that encodes the preconditions for executing each task of the process using the space-optimized data structure.
Pouheidari et al. [22]	Investigating a case study of Execution of Untrusted Business Process on Hyperledger Composer [79] and Hyperledger Fabric [6] as a permissioned blockchain and investigate the advantages of using permissioned blockchain for the collaborative business process.
Mendlin et al. [23]	A survey paper that investigates challenges and opportunities of blockchain for BPM.
López-Pintado et al. [24]	A blockchain-based Business Process Management System. It uses Solidity smart contracts to encode the state of the business process instance and execute workflow routing.
Ribma et al.,[25]	Comparing the cost of computation and storage of business process execution on blockchain and cloud platforms. They represent a cost model based on the cost of deploying smart contracts and executing process coordination.

Caterpillar [24] is also BPM system based on Ethereum blockchain. They represent a compiler for compiling BPMN to Solidity contracts. It supports a variety of BPMN constructs such as user, script and service tasks, parallel, exclusive, and event-based gateways, and a lot more in the new version.[21]

Table 14 summarizes BPM studies with the focus on smart contract.

E. RECORD KEEPING

One of the main application of blockchain is considering it as a secure, tamper-proof and trustable database for record keeping. In many applications, this characteristic of the blockchain has been remarked, such as IOT, Supply Chain, Health care, and BPM. Also, using blockchain technology and smart contract has been considered in research studies merely for record keeping applications as well. Table 15 presents the summary of these studies.

Lemieux [42] investigate the potential of blockchain for the record keeping purpose by discussing the problems regarding records preserving and reliability because of the lack of a Trusted Digital Repository and the advantages such as the low-cost transaction fee.

D'Angelo *et al.* [39] consider employing smart contracts and blockchain technology alongside with could infrastructure for the data recording purpose in order to address the accountability and trust issues in cloud platforms. They assert three different execution architecture. One of the presented architecture is ''blockchain-based logging with smart contracts''. In this architecture, smart contracts act as an arbitrator. Smart contracts are responsible for event verification, Service Level Agreements (SLA) violations, and calculating fines. So the disputations are resolved automatically based on the smart contract regulations.

Lemieux [41] conducts a typology for the blockchain-based record keeping solutions include three main design patterns mirror type, digital record type, and tokenized type. Mirror type uses blockchain as a repository of hashes of original record to guarantee the safety of the records and preserving records from any undesirable changes. This method has been implemented by both permissioned and public Blockchains. Digital record types are more than just a repository, they apply smart contracts to regulate the records and create an active and dynamic record keeping system. Tokenized type are the most advanced type, which record assets on the Blockchain beside the regular records. Assets can be indicative of any valuable good and can be represented by a token or cryptocurrency and record on the blockchain.

Lemieux also introduces a science-based theoretical framework for evaluating blockchain-based record keeping systems [40]. She expresses that a trustworthy record has three characteristics, accuracy, reliability, and authenticity. The blockchain tamper-proof characteristic that captures records as a chain of hashes can guarantee these attributes. Reliability requires three preconditions, completeness at the point of the creation, consistency with the formal rules, and naturalness. Also, there are two preconditions for authenticity: identity and integrity to detect genuine records from forgery records.

F. VOTING

Blockchain can help to create a secure and privacy-preserving voting system. Generally, voting systems suffer from trust issues and the chance of cheating and leakage is remarkable in centralized voting systems. Table 16 presents voting system based on smart contracts.

McCorry *et al.* [44] present a privacy-preserving boardroom voting system using smart contacts and Zero-knowledge proof protocol. The implementation is based on Solidity smart contracts and Ethereum public blockchain. The two main smart contracts are ''voting contract'' and ''cryptography contract''. The voting contract implements the voting protocol and verifies zero-knowledge proofs. The cryptography contract is responsible for generating two type of zero-knowledge proofs for the voters: Schnorr proof [130] and one-out-of-two proof [131].

Shah also proposes a voting system, which integrates client-server architecture with blockchain [45]. The four main components of the proposed system include User or front-end interface, Authentication server, Arbitration Server, and private Blockchain. The smart contract is responsible for

[21] https://github.com/orlenyslp/Caterpillar

TABLE 15. Record keeping decentralized application.

Reference	Application feature
DÁngelo et al. [39]	Using blockchain-based logging system to log and record the interactions and employing smart contract as an arbitrator to verify the stored events, detecting (Service Level Agreements) SLA violations and calculating compensations.
Lemieux [40]	A theoretic framework to evaluate the potential of the blockchain and smart contract as a decentralized trusted record keeping system.
Lemieux [41]	Creating and updating records on blockchain using smart contracts by encoding procedures that run on a multi-stakeholder network.
Guo et al. [43]	Using blockchain as an event recording system for autonomous vehicles. A "proof of Event" mechanism with Dynamic Federation consensus records is suggested to resolve accidents disputes.

TABLE 16. Voting decentralized application.

Reference	Application feature
McCorry et al. [44]	Completely untrusted protocol based on smart contract for boardroom voting that guarantees voters privacy. The smart contracts handle the voting protocol, monitor the election process, create and verify the zero-knowledge proofs [129].
Shah [45]	A blockchain-based voting system that uses smart contract codes for verifying and adding a record of votes to the blockchain.
Chen et al. [46]	Using smart contract to solve the problem of the bid price leakage before the specified deadline. The smart contract includes the auctioneer's address, start time, deadline, address of the current winner and current highest offer.

verifying the votes based on the type of the election (interim or non-interim) and adding the vote as a new record on the Blockchain.

Chen *et al.* [46] address the bid price leakage in the bidding system by proposing an approach using smart contracts implemented in Solidity and Ethereum blockchain. The five functions implemented by the auction smart contract are: blindAuction(), bid(), reveal(), auctionEnd(), and withdraw(). The blindAuction activates the contract and records the start time and end time. The bid function calls by anyone who wants to send the bid if the deadline does not expire. The reveal function reveals the bid prices. The auctionEnd checks the bid deadline to define the ending of the auction and selecting the winner. Finally, the withdraw function returns the bids tendered of the losers.

G. DIGITAL IDENTITY

Digital identities provided by centralized solutions raise several problems such as users privacy and untrusted third parties. Moreover, usually they cannot provide a single identity to be adopted for different organizations. Consequently, users' private information could be accessible to many third parties that handle digital identities. Blockchain and smart contract can contribute in digital identity systems to address these issues [36]. Bendiab *et al.* [38] similarly recommend using blcokchain for cloud identity management system to address the trust issue in a dynamic and distributed approach.

Mühle *et al.* present four essential components of self-sovereign identity (SSI) [37]. Self-sovereign is a concept that users can manage their own identities and the IDs can be validated without centralized parties. The four main represented components include identification, authentication, verification, and storage. For identification component, the paper discusses how different blockchain platforms such as Ethereum and Bitcoin generate an identifier for identities and how decentralized identity systems such as uPort utilize it. Authentication component uses Public Key Infrastructure (PKI) and Zero-knowledge algorithm to authenticate the user identity. The verification component denotes the process of raising the claim, which could have multiple attestation. The paper also discusses two main applied methods for linking claims and attestations, ''Identity Registry'' and ''Claim Registry''. Lastly, the storage component discusses on-chain and off-chain solutions and their pros and cons in the context of privacy and scalability.

Al-Bassam [34] presents an SCPKI system (Smart Contract Public Key Infrastructure). The public keys and attributes of identity record on the blockchain and the smart contracts are applied to control and manage them. The study introduces two design models for smart contracts. In the first design (full version), all identity information such as attributes, signatures, and revocations are stored in an array of attributes and can be accessible to other smart contracts. The second one is a lighter design, which the other smart contracts do not have access to all identity attributes of a specific smart contract and they just have access to the attribute ID. The lighter version consumes less GAS, therefore, less cost.

DeCusatis and Sager [35] present a test result based on cloud infrastructure, BlackRidge [22] technology, Hyperledger Fabric blockchain, and chaincode (Hyperledger Fabric smart contract) to manage digital identities. The implementation is based on two BlackRidge services: BlackRidge First Packet Authentication and BlackRidge Transport Access Control (TAC). Network segmentation and traffic separation have been suggested to enable multiple organization share the same blockchain infrastructure to provide precise compliance audit. The BlackRidge software generates identity in client-side and inserts it into the first packet heather and it is authenticated by the BlackRidge endpoint installed on Hyperledger fabric server to secure blockchain from unauthorized access.

Yasin and Liu propose a framework that aggregates online identities and uses smart contract for ranking the reputation of the users [33].

[22]https://www.blackridge.us/

TABLE 17. Digital identity decentralized application.

References	Application Feature
Yasin and Liu [33]	A framework for online identity that uses smart contracts to calculate personal online ratings based on the aggregated data from all the digital identities of the user.
Al-Bassam [34]	Using web-of-trust and smart contract to manage identities and their attributes that stored on the blockchain and create a decentralized PKI and identity system.
DeCusatis and Sager [35]	Investigating a prototype that implements an end-to-end user identity management system using Hyperledger Fabric. Suggesting identity-based network segmentation and traffic separation to increase the security and network tolerance against DDoS attack.
Grüner et al. [36]	First, it discusses the decision of using blockchain based digital identity system. Second, it analyzes uPort[132] Sovrin[133], and ShoCard[134]. The analysis result indicates uPort can be best fit with blockchain technology. Strict trust framework of Sovrin may not work with blockchain and decentalized nature.
Mühle et al. [37]	Discussing the main components, architecture and actors of a self-sovereign identity management system based on blockchain and smart contracts as well as authentication and storage solutions.

Table 17 presents the summary of the studies related to the digital identity decentralized applications and smart contracts.

VI. CONCLUSIONS, CHALLENGES AND FUTURE DIRECTIONS

This paper studied smart contract as a key component of distributed ledger technology. To our knowledge, this is the first study on the smart contract topic. In this research study, we provided a systematic review on the smart contract history, supporting platforms, programming languages, security, performance, and decentralized applications. We systematically searched for papers from different online databases based on the designed research questions and finally selected 90 papers based on their relevance and quality.

In order to tackle practical and competitive decentralized applications, improvements toward the security and performance of smart contracts are required. We have provided a classification of different security approaches for detecting smart contract vulnerabilities and examined presented tools and their implementation. Also, we have introduced different approaches to optimize the performance of smart contract execution and run the smart contract transactions concurrently. Performance analysis frameworks for measuring the performance of different blockchain platforms is highlighted as well.

Based on our wide survey of decentralized applications, we have presented seven different categories for smart contract-based applications. The problems that each category addresses and tries to solve has been discussed, as well as implementations details with the focus on smart contract structure and purpose. The significant interest in applying decentralized ledger technology in various fields demonstrates that we can expect to see a demanding trend for more variety of business markets and more areas based on this novel technology, specifically along with cloud services.

Access control management, sharing resources, and record keeping are the primary focus of the decentralized applications specifically in the fields of the healthcare, and IoT, so despite the verity of the applications their focus is narrow down to these three aspects.

Ethereum and Solidity are currently the dominant platform and programming language so far. We expect that in future more research studies focus on permissioned platforms such as Hyperledger Fabric, Quorum, Corda, and Tendermint because of their desirable features for enterprise applications. Also, most of the proposed solutions are dependent on a specific platform and are not applicable as a generic solution.

Many proposed applications require integration between on-chain and off-chain. Although some studies recommended using SGX to overcome the potential security problems, the other possible challenges such as cache attacks [135] are not investigated and requires further studies.

In order to improve the performance of the applications based on distributed ledger technology two main approaches recognized, using lighter consensus mechanism, and running transactions concurrently. However, the performance of the blockchain-based solutions only compared with other blockchain solutions and still, there is a big gap between the performance of the blockchain-based solution and current applications. Research toward improving the performance of executing smart contracts and the overall blockchain-based applications are still in early stages. A lot of research needs to be done to fill this gap and make blockchain-based applications competitive in real markets.

REFERENCES

[1] S. Nakamoto. (2017). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf
[2] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *Proc. 2nd Int. Conf. Open Big Data (OBD)*, Aug. 2016, pp. 25–30.
[3] Q. I. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "MeDShare: Trust-less medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14757–14767, 2017.
[4] Q. Xia, E. B. Sifah, A. Smahi, S. Amofa, and X. Zhang, "BBDS: Blockchain-based data sharing for electronic medical records in cloud environments," *Information*, vol. 8, no. 2, p. 44, 2017.
[5] S. Rouhani, L. Butterworth, A. D. Dimmond, D. G. Humphery, and R. Deters, "Medichaintm: A secure decentralized medical data asset management system," in *2018 IEEE Conf. Internet Things, Green Comput. Commun., Cyber, Phys. Social Comput., Smart Data, Blockchain, Comput. Inf. Technol., Congr. Cybermatics*, Aug. 2018, pp. 14757–14767.
[6] T. Mikula and R. H. Jacobsen, "Identity and access management with blockchain in electronic healthcare records," in *Proc. 21st Euromicro Conf. Digital Syst. Design (DSD)*, Aug. 2018, pp. 699–706.

[7] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control," *J. Med. Syst.*, vol. 40, no. 10, p. 218, 2016.

[8] K. Peterson, R. Deeduvanu, P. Kanjamala, and K. Boles, "A blockchain-based approach to health information exchange networks," in *Proc. NIST Workshop Blockchain Healthcare*, vol. 1, 2016, pp. 1–10.

[9] A. Theodouli, S. Arakliotis, K. Moschou, K. Votis, and D. Tzovaras, "On the design of a blockchain-based system to facilitate healthcare data sharing," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./12th IEEE Int. Conf. Big Data Sci. Eng.*, Aug. 2018, pp. 1374–1379.

[10] T. Nugent, D. Upton, and M. Cimpoesu, "Improving data transparency in clinical trials using blockchain smart contracts," NCBI, U.S. Nat. Library Med., Tech. Rep. F1000Research, vol. 5, 2016.

[11] Z. Shae and J. J. P. Tsai, "On the design of a blockchain platform for clinical trial and precision medicine," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 1972–1980.

[12] P. Zhang, J. White, D. C. Schmidt, G. Lenz, and S. T. Rosenbloom, "FHIRChain: Applying blockchain to securely and scalably share clinical data," *Comput. Struct. Biotechnol. J.*, vol. 16, pp. 267–278, Jul. 2018. doi: 10.1016/j.csbj.2018.07.004.

[13] P. Zhang, M. A. Walker, J. White, D. C. Schmidt, and G. Lenz, "Metrics for assessing blockchain-based healthcare decentralized apps," in *Proc. 19th Int. Conf. e-Health Netw., Appl. Services (Healthcom)*, Oct. 2017, pp. 1–4.

[14] S. Chen, R. Shi, Z. Ren, J. Yan, Y. Shi, and J. Zhang, "A blockchain-based supply chain quality management framework," in *Proc. IEEE 14th Int. Conf. e-Bus. Eng. (ICEBE)*, Nov. 2017, pp. 172–176.

[15] T. Bocek, B. B. Rodrigues, T. Strasser, and B. Stiller, "Blockchains everywhere—A use-case of blockchains in the pharma supply-chain," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 772–777.

[16] K. Korpela, J. Hallikas, and T. Dahlberg, "Digital supply chain transformation toward blockchain integration," in *Proc. 50th Hawaii Int. Conf. Syst. Sci.*, 2017, pp. 1–5.

[17] M. Ruta, F. Scioscia, S. Ieva, G. Capurso, and E. Di Sciascio, "Supply chain object discovery with semantic-enhanced blockchain," in *Proc. 15th ACM Conf. Embedded Netw. Sensor Syst.*, Nov. 2017, p. 60.

[18] H. M. Kim and M. Laskowski, "Toward an ontology-driven blockchain design for supply-chain provenance," *Intell. Syst. Accounting, Finance Manage.*, vol. 25, no. 1, pp. 18–27, Jan. 2018.

[19] S. Figorilli *et al.*, "A blockchain implementation prototype for the electronic open source traceability of wood along the whole supply chain," *Sensors*, vol. 18, no. 9, p. 3133, Sep. 2018.

[20] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling, "Untrusted business process monitoring and execution using blockchain," in *Business Process Management*. New York, NY, USA: Springer, 2016, pp. 329–347.

[21] L. García-Bañuelos, A. Ponomarev, M. Dumas, and I. Weber, "Optimized execution of business processes on blockchain," in *Business Process Management*. New York, NY, USA: Springer, 2017, pp. 130–146.

[22] V. Pourheidari, S. Rouhani, and R. Deters, "A case study of execution of untrusted business process on permissioned blockchain," in *Proc. IEEE Confs Internet Things, Green Comput. Commun., Cyber, Phys. Social Comput., Smart Data, Blockchain, Comput. Inf. Technol., Congr. Cybermatics*, Aug. 2018, pp. 1588–1594.

[23] J. Mendling *et al.*, "Blockchains for business process management-challenges and opportunities," *ACM Trans. Manage. Inf. Syst. (TMIS)*, vol. 9, no. 1, p. 4, Feb. 2018.

[24] O. Löpez-Pintado, L. García-Bañuelos, M. Dumas, and I. Weber, "Caterpillar: A blockchain-based business process management system," Ph.D. dissertation, BPM, Barcelona, Spain, 2017.

[25] P. Rimba, A. B. Tran, I. Weber, M. Staples, A. Ponomarev, and X. Xu, "Comparing blockchain and cloud services for business process execution," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Apr. 2017, pp. 257–260.

[26] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "FairAccess: A new Blockchain-based access control framework for the Internet of Things," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5943–5964, 2017.

[27] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "Towards a novel privacy-preserving access control model based on blockchain technology in IoT," in *Proc. Europe MENA Cooperation Adv. Inf. Commun. Technol.*, 2017, pp. 523–533.

[28] R. Xu, Y. Chen, E. Blasch, and G. Chen. (2018). "Blendcac: A blockchain-enabled decentralized capability-based access control for iots." [Online]. Available: https://arxiv.org/abs/1804.09267.

[29] S.-C. Cha, J.-F. Chen, C. Su, and K.-H. Yeh, "A blockchain connected gateway for ble-based devices in the Internet of Things," *IEEE Access*, vol. 6, pp. 24639–24649, 2018.

[30] S. Huh, S. Cho, and S. Kim, "Managing IoT devices using blockchain platform," in *Proc. 19th Int. Conf. Adv. Commun. Technol.*, Feb. 2017, pp. 464–467.

[31] M. Ruta, F. Scioscia, S. Ieva, G. Capurso, A. Pinto, and E. D. Sciascio, "A blockchain infrastructure for the semantic web of things," in *Proc. 26th Italian Symp. Adv. Database Syst.*, Aug. 2018, pp. 258-269.

[32] F. Lombardi, L. Aniello, S. de Angelis, A. Margheri, and V. Sassone, "A blockchain-based infrastructure for reliable and cost-effective IoT-aided smart grids," in *Proc. Living Internet Things Cybersecurity IoT*, 2018, pp. 1–6.

[33] A. Yasin and L. Liu, "An online identity and smart contract management system," in *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jun. 2016, pp. 192–198.

[34] M. Al-Bassam, "Scpki: A smart contract-based PKI and identity system," in *Proc. ACM Workshop Blockchain, Cryptocurrencies Contracts*, Apr. 2017, pp. 35–40.

[35] C. DeCusatis, M. Zimmermann, and A. Sager, "Identity-based network security for commercial blockchain services," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 474–477.

[36] A. Grüner, A. Mühle, and C. Meinel. (2018). "On the relevance of blockchain in identity management." [Online]. Available: https://arxiv.org/abs/1807.08136

[37] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, "A survey on essential components of a self-sovereign identity," *Comput. Sci. Rev.*, vol. 30, pp. 80–86, Nov. 2018.

[38] K. Bendiab, N. Kolokotronis, S. Shiaeles, and S. Boucherkha, "WiP: A novel blockchain-based trust model for cloud identity management," in *Proc. IEEE 16th Intl. Conf. Dependable, Autonomic Secure Comput.*, Aug. 2018, pp. 724–729.

[39] G. D'Angelo, S. Ferretti, and M. Marzolla. (2018). "A blockchain-based flight data recorder for cloud accountability." [Online]. Available: https://arxiv.org/abs/1806.04544

[40] V. L. Lemieux, "Blockchain and distributed ledgers as trusted recordkeeping systems," in *Proc. Future Technol. Conf. (FTC)*, 2017, pp. 41–48.

[41] L. L. Victoria, "A typology of blockchain recordkeeping solutions and some reflections on their implications for the future of archival preservation," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 2271–2278.

[42] V. L. Lemieux, "Trusting records: Is blockchain technology the answer?" *Records Manage. J.*, vol. 26, no. 2, pp. 110–139, Jul. 2016.

[43] H. Guo, E. Meamari, and C.-C. Shen, "Blockchain-inspired event recording system for autonomous vehicles," in *Proc. 1st IEEE Int. Conf. Hot Inf.-Centric Netw. (HotICN)*, Aug. 2018, pp. 218–222.

[44] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Proc. Int. Conf. Financial Cryptogr. Data Security*. Cham, Switzerland: Springer, 2017, pp. 357–375.

[45] S. Shah. *Block Chain Voting System*. [Online]. Available: https://www.economist.com/sites/default/files/northeastern.pdf

[46] Y.-H. Chen, S.-H. Chen, and I.-C. Lin, "Blockchain based smart contract for bidding system," in *Proc. IEEE Int. Conf. Appl. Syst. Invention (ICASI)*, Apr. 2018, pp. 208–211.

[47] S. Gec, D. Lavbič, M. Bajec, and V. Stankovski. (2018). "Smart contracts for container based video conferencing services: Architecture and implementation." [Online]. Available: https://arxiv.org/abs/1808.03832

[48] S. A. Suchaad *et al.*, "Blockchain use in home automation for children incentives in parental control," in *Proc. Int. Conf. Mach. Learn. Mach. Intell.*, Feb. 2018, pp. 50–53.

[49] S. Rouhani, V. pourheidari, and R. Deters, "Physical access control management system based on permissioned blockchain," in *Proc. IEEE Conf. Internet Things, Green Comput. Commun., Cyber, Phys. Social Comput., Smart Data, Blockchain, Comput. Inf. Technol., Congr. Cybermatics*, Jul. 2018, pp. 1078–1083.

[50] K. Salah, M. Rehman, N. Nizamuddin, and A. Al-Fuqaha, "Blockchain for AI: Review and open research challenges," *IEEE Access*, vol. 7, pp. 10127–10149, 2019.

[51] S. Nick. (2017). *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
[52] S. Nick. (1997). *Formalizing and Securing Relationship on Public Networks*. [Online]. Available: http://firstmonday.org/ojs/index.php/fm/article/view/548/469
[53] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology?—A systematic review," *PloS One*, vol. 11, no. 10, 2016, Art. no. e0163477.
[54] Z. Zheng, S. Xie, H.-N. Dai, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2016.
[55] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (SoK)," in *Principles Security Trust*. New York, NY, USA: Springer, 2017, pp. 164–186.
[56] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. Leung, "Decentralized applications: The blockchain-empowered software system," *IEEE Access*, vol. 6, pp. 53019–53033, 2018.
[57] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics Inform.*, vol. 36, pp. 55–81, Mar. 2018.
[58] T.-T. Kuo, H.-E. Kim, and L. Ohno-Machado, "Blockchain distributed ledger technologies for biomedical and health care applications," *J. Amer. Med. Inform. Assoc.*, vol. 24, no. 6, pp. 1211–1220, 2017.
[59] M. Hölbl, M. Kompara, A. Kamišalić, and L. N. Zlatolas, "A systematic review of the use of blockchain in healthcare," *Symmetry*, vol. 10, no. 10, p. 470, Oct. 2018.
[60] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, "Blockchain and IoT integration: A systematic survey," *Sensors*, vol. 18, no. 8, p. 2575, 2018.
[61] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, Jan. 2009.
[62] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," *Ease*, vol. 8, pp. 68–77, Jun. 2008.
[63] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, p. 30.
[64] W. Wang *et al.* (2018). "A survey on consensus mechanisms and mining management in blockchain networks." [Online]. Available: https://arxiv.org/abs/1805.02707
[65] M. Bartoletti and R. Zunino, "Bitml: A calculus for bitcoin smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, Oct. 2018, pp. 83–100.
[66] F. Schrans, S. Eisenbach, and S. Drossopoulou, "Writing safe smart contracts in flint," in *Proc. Conf. Companion 2nd Int. Conf. Art, Sci., Eng. Program.*, Apr. 2018, pp. 218–219.
[67] I. Sergey, A. Kumar, and A. Hobor. (2018). "Scilla: A smart contract intermediate-level language." [Online]. Available: https://arxiv.org/abs/1801.00687
[68] G.-T. Nguyen and K. Kim, "A survey about consensus algorithms used in blockchain," *J. Inf. Process. Syst.*, vol. 14, no. 1, pp. 101–128, 2018.
[69] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology—EUROCRYPT*. New York, NY, USA: Springer, 2015, pp. 281–310.
[70] F. Schuh and D. Larimer. (2017). *Bitshares 2.0: General Overview*. [Online]. Available: http://docs.bitshares.org/_downloads/bitshares-general. pdf
[71] (2019). *Nem Technical Reference*. [Online]. Available: https://nem.io/wp-content/themes/nem/files/NEM_techRef.pdf
[72] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.
[73] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 305–319.
[74] (2017). *Istanbul Byzantine Fault Tolerant Consensus Protocol*. [Online]. Available: https://github.com/ethereum/EIPs/issues/650
[75] P. L. Seijas, S. J. Thompson, and D. McAdams, "Scripting smart contracts for distributed ledger technology," IACR Cryptol. ePrint Arch., Tech. Rep. 2016 1156, 2016.
[76] C. Dannen, *Introducing Ethereum Solidity*. New York, NY, USA: Springer, 2017.
[77] R. Zubairy. (2019). *Create a Blockchain App for Loyalty Points With Hyperledger Fabric Ethereum Virtual Machine*. [Online]. Available: https://developer.ibm.com/patterns/loyalty-points-fabric-evm/
[78] V. Buterin. (2016). *Critical Update Re: Dao Vulnerability Ethereum Blog*. [Online]. Available: https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability/
[79] F. Idelberger, G. Governatori, R. Riveret, and G. Sartor, "Evaluation of logic-based smart contracts for blockchain systems," in *Rule Technologies. Research, Tools, and Applications*. New York, NY, USA: Springer, 2016, pp. 167–183.
[80] H.-P. Lam and G. Governatori, "The making of SPINdle," in *Rule Interchange and Applications*. New York, NY, USA: Springer, 2009, pp. 315–322.
[81] X. He, B. Qin, Y. Zhu, X. Chen, and Y. Liu, "Spesc: A specification language for smart contracts," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2018, pp. 132–137.
[82] Y. Hirai. (2018). *Bamboo: A Language for Morphing Smart Contracts*. [Online]. Available: https://github.com/pirapira/bamboo
[83] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, "Step by step towards creating a safe smart contract: Lessons and insights from a Cryptocurrency lab," in *Financial Cryptography and Data Security*. New York, NY, USA: Springer, 2016, pp. 79–94.
[84] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, Oct. 2016, pp. 254–269.
[85] K. Bhargavan *et al.*, "Formal verification of smart contracts: Short paper," in *Proc. ACM Workshop Program. Lang. Anal. Security*, Oct. 2016, pp. 91–96.
[86] N. Swamy *et al.*, "Dependent types and multi-monadic effects in f," *ACM SIGPLAN Notices*, vol. 51, no. 1, pp. 256–270, Jan. 2016.
[87] S. Amani and M. Bégel, M. Bortin, and M. Staples, "Towards verifying ethereum smart contract bytecode in Isabelle/HOL," in *Proc. 7th ACM SIGPLAN Int. Conf. Certified Programs Proofs*, Jan. 2018, pp. 66–77.
[88] D. Park, Y. Zhang, M. Saxena, P. Daian, and G. Roşu, "A formal verification tool for ethereum vm bytecode," in *Proc. 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2018, pp. 912–915.
[89] E. Hildenbrandt *et al.*, "Kevm: A complete semantics of the ethereum virtual machine," in *Proc. IEEE 31st Comput. Secur. Found. Symp. (CSF)*, Oxford, U.K., Jul. 2018, pp. 204–217.
[90] E. Albert, P. Gordillo, B. Livshits, A. Rubio, and I. Sergey. (2018). "Ethir: A framework for high-level analysis of ethereum bytecode." [Online]. Available: https://arxiv.org/abs/1805.07208
[91] E. Albert, P. Gordillo, A. Rubio, and I. Sergey. (2018). "Gastap: A gas analyzer for smart contracts." [Online]. Available: https://arxiv.org/abs/1811.10403
[92] S. Bragagnolo, H. Rocha, M. Denker, and S. Ducasse, "SmartInspect: Solidity smart contract inspector," in *Proc. Int. Workshop Blockchain Oriented Softw. Eng. (IWBOSE)*, Mar. 2018, pp. 9–18.
[93] P. Tsankov, A. Dan, D. D. Cohen, A. Gervais, F. Buenzli, and M. Vechev. (2018). "Securify: Practical security analysis of smart contracts." [Online]. Available: https://arxiv.org/abs/1806.01143
[94] I. Nikolic, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor. (2018). "Finding the greedy, prodigal, and suicidal contracts at scale." [Online]. Available: https://arxiv.org/abs/1802.06038
[95] L. Brent *et al.* (2018). "Vandal: A scalable security analysis framework for smart contracts." [Online]. Available: https://arxiv.org/abs/1809.03981
[96] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov, "SmartCheck: Static analysis of ethereum smart contracts," in *Proc. IEEE/ACM 1st Int. Workshop Emerg. Trends Softw. Eng. Blockchain (WETSEB)*, Jun. 2018, pp. 9–16.
[97] T. Chen, X. Li, X. Luo, and X. Zhang, "Under-optimized smart contracts devour your money," in *Proc. IEEE 24th Int. Conf. Softw. Anal., Evol. Reengineering (SANER)*, Feb. 2017, pp. 442–446.
[98] S. Grossman *et al.*, "Online detection of effectively callback free objects with applications to smart contracts," *ACM Program. Lang.*, vol. 2, p. 48, Dec. 2017.
[99] M. Fröwis and R. Böhme, "In code we trust?—Measuring the control flow immutability of all smart contracts deployed on ethereum," in *Proc. Int. Workshops Data Privacy Manage., Cryptocurrencies Blockchain Technol. ESORICS DPM/CBT@ESORICS*, Oslo, Norway, Springer, 2017.
[100] H. Watanabe, S. Fujimura, A. Nakadaira, Y. Miyazaki, A. Akutsu, and J. Kishigami, "Blockchain contract: Securing a blockchain applied to smart contracts," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2016, pp. 467–468.

[101] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, Oct. 2016, pp. 270–282.
[102] A. Kothapalli, A. Miller, and N. Borisov, "SmartCast: An incentive compatible consensus protocol using smart contracts," in *Int. Conf. Financial Cryptogr. Data Secur.*, 2017, pp. 536–552.
[103] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Security Privacy (SP)*, May 2016, pp. 839–858.
[104] G. Zyskind, O. Nathan, and A. Pentland. (2015). "Enigma: Decentralized computation platform with guaranteed privacy." [Online]. Available: https://arxiv.org/abs/1506.03471
[105] G. Bracha and D. M. Ungar, "Mirrors: Design principles for meta-level facilities of object-oriented programming languages," in *Proc. OOPSLA*, Vancouver, BC, Canada, 2004, pp. 331–344.
[106] N. Papoulias, "Remote debugging and reflection in resource constrained devices," Ph.D. dissertation, Dept. Comput. Sci., Univ. Sci. Technol. Lille-Lille I, Villeneuve-d'Ascq, France, 2013.
[107] E. Albert *et al.*, "Saco: Static analyzer for concurrent objects," in *Proc. Int. Conf. Tools Algorithms Construct. Anal. Syst.*, 2014, pp. 562–567.
[108] E. Albert, P. Arenas, S. Genaim, and G. Puebla, "Automatic inference of upper bounds for recurrence relations in cost analysis," in *Proc. Int. Static Anal. Symp.*, 2008, pp. 221–237.
[109] H. Jordan, B. Scholz, and P. Subotić, "Soufflè: On synthesis of program analyzers," in *Proc. Int. Conf. Comput. Aided Verification*, 2016, pp. 422–430.
[110] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," Ethereum, Zug, Switzerland, White Paper, 2014.
[111] A. Stefănescu, D. Park, S. Yuwen, Y. Li, and G. Roşu, "Semantics-based program verifiers for all languages," *ACM SIGPLAN Notices*, vol. 51, no. 10, pp. 74–91, 2016.
[112] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014, [Online]. Available: http://gavwood.com/paper.pdf
[113] A. Clement, H. C. Li, J. Napper, J.-P. Martin, L. Alvisi, and M. Dahlin, "Bar primer," in *Proc. DSN*, vol. 8. Jun. 2008, pp. 287–296.
[114] C. Molina-Jimenez, E. Solaiman, I. Sfyrakis, I. Ng, and J. Crowcroft. (2018). "On and off-blockchain enforcement of smart contracts." [Online]. Available: https://arxiv.org/abs/1805.00626
[115] R. Cheng *et al.*, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution," *arXiv preprint arXiv:1804.05141*, 2018.
[116] M. Herlihy and E. Koskinen, "Transactional boosting: A methodology for highly-concurrent transactional objects," in *Proc. 13th ACM SIGPLAN Symp. Principles Pract. Parallel Programming*, 2008, pp. 207–216.
[117] T. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, "Adding concurrency to smart contracts," in *Proc. ACM Symp. Principles Distrib. Comput.*, Jul. 2017, pp. 303–312.
[118] L. Yu, W.-T. Tsai, G. Li, Y. Yao, C. Hu, and E. Deng, "Smart-contract execution with concurrent block building," in *Proc. 11th IEEE Symp. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2017, pp. 160–167.
[119] P. S. Anjana, S. Kumari, S. Peri, S. Rathor, and A. Somani. (2018). "An efficient framework for concurrent execution of smart contracts." [Online]. Available: https://arxiv.org/abs/1809.01326
[120] N. Shavit and D. Touitou, "Software transactional memory," *Distrib. Comput.*, vol. 10, no. 2, pp. 99–116, 1997.
[121] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proc. ACM Int. Conf. Manage. Data*, May 2017, pp. 1085–1100.
[122] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proc. 1st ACM Symp. Cloud Comput.*, Jun. 2010, pp. 143–154.
[123] M. J. Cahill, U. Röhm, and A. D. Fekete, "Serializable isolation for snapshot databases," *ACM Trans. Database Syst.*, vol. 34, no. 4, p. 20, Dec. 2009.
[124] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A detailed and real-time performance monitoring framework for blockchain systems," in *Proc. 40th Int. Conf. Softw. Eng. Softw. Eng. Pract.*, Jun. 2018, pp. 134–143.
[125] M. Samaniego and R. Deters, "Blockchain as a service for IoT," in *Proc. IEEE Int. Conf. Internet Things (iThings)*, Dec. 2016, pp. 433–436.
[126] M. Samaniego and R. Deters, "Hosting virtual IoT resources on edge-hosts with blockchain," in *Proc. IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, Dec. 2016, pp. 116–119.
[127] K. Korpela, U. Kuusiholma, O. Taipale, and J. Hallikas, "A framework for exploring digital business ecosystems," in *Proc. 46th Hawaii Int. Conf. Syst. Sci.*, Jan. 2013, pp. 3838–3847.
[128] B. Clegg and B. Tan, "Using qfd for e-business planning and analysis in a micro-sized enterprise," *Int. J. Qual. Rel. Manage.*, vol. 24, no. 8, pp. 813–828, 2007.
[129] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "SNARKs for C: Verifying program executions succinctly and in zero knowledge," in *Advances in Cryptology–CRYPTO*. New York, NY, USA: Springer, 2013, pp. 90–108.
[130] C. P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991.
[131] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Proc. Annu. Int. Cryptol. Conf.*, 1994, pp. 174–187.
[132] C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena. (2018). *Uport: A Platform for Self-Sovereign Identity*. [Online]. Available: http://blockchainlab.com/pdf uPort_whitepaper_DRAFT20161020.pdf
[133] D. Reed, J. Law, and D. Hardman. (2018). *The Technical Foundations of Sovrin A White Paper From the Sovrin Foundation*. [Online]. Available: https://www.evernym.com/wp-content/uploads/2017/07/The-Technical-Foundations-of-Sovrin.pdf
[134] shocard. (2017). *Shocard With Shocoin Tokens Whitepaper Identity Management Verified Using the Blockchain*. [Online]. Available: http://www.lianzhiliao.com/media/whitepapers/ShoCard-Whitepaper_en.pdf
[135] F. Brasser and U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A.-R. Sadeghi, "Software grand exposure:$*SGX*$ cache attacks are practical," in *Proc. 11th USENIX Workshop Offensive Technol.*, Aug. 2017, pp. 1–10.

SARA ROUHANI is currently pursuing the Ph.D. degree in computer science with the University of Saskatchewan, Canada, under the supervision of Prof. R. Deters. She is also a Research Assistant with the Multi-User Adaptive Distributed Mobile and Ubiquitous Computing (MADMUC) Laboratory led by Prof. R. Deters and Prof. J. Vassileva. Her research interests include distributed systems, blockchain, and smart contracts.

RALPH DETERS received the Ph.D. degree from Federal Armed Forces University, Munich, Germany, in 1998. He joined the University of Saskatchewan as a Research Associate, in 1998, where he is currently a Full Professor with the Department of Computer Science. His research interests include distributed ledger technology, the Internet of Things, and cloud/edge computing.