Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technologies

Bachelor Thesis

Blockchain infrastructure in e-Governance and its Data Security

Yoseph Zeleke

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Yoseph Fekadu Zeleke

Systems Engineering and Informatics
Informatics

Thesis title

Block-chain infrastructure in e-Governance and its Data Security

---

Objectives of thesis

The main aim of this bachelor's thesis is to delve into the implementation of several applications of Blockchain Technology into E-Government and its cybersecurity in data protection. Other side goals are the description of the Blockchain infrastructure as a service.

Methodology

The methodology of this bachelor's thesis is based on the study and gathering of information on the topic from the literature, the analysis of professional information sources and case studies, and the comparative method of applications of the technology. It will put emphasis on the architecture, application, and security of the Blockchain network. Subsequently, it will put conclusions justifying the objectives of this thesis.

The proposed extent of the thesis

60 pages

Keywords

Blockchain, Cryptocurrency, Cryptography, Decentralization, e-Government

Recommended information sources

KOSHIK, Raj : Foundations of Blockchain : The Pathway to Cryptocurrencies and Decentralized Blockchain Applications, 2019. ISBN : 978-1-78913-939-6

RAVAL, Siraj : Decentralized Applications: Harnessing Bitcoin's Blockchain Technology, 2016. ISBN : 978-1-491-92454-9

VEUGER, Jan : Blockchain technology and applications, 2019. ISBN : 978-1-53615-289-0

Expected date of thesis defence

2021/22 SS – FEM

The Bachelor Thesis Supervisor

Ing. Tomáš Vokoun

Supervising department

Department of Information Technologies

Electronic approval: 29. 7. 2020

Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 19. 10. 2020

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 14. 03. 2022

Declaration

I declare that I have worked on my bachelor thesis titled “Blockchain infrastructure in e-Governance and its data security” by myself, and I have used only the sources mentioned at the end of the thesis. Furthermore, as the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on 15.03.2022

Acknowledgment

I would like to thank my thesis supervisor Ing. Tomáš Vokoun, for the useful comments, remarks, and engagement through the learning process of this bachelor thesis. Furthermore, I would like to express my profound gratitude to my family and friends for providing me with unfailing support and continuous encouragement throughout my years of study and writing this thesis.

Abstract

In the current information culture, simple, convenient, and effective engagement between government and citizens has become a common expectation. Electronic government solutions, which are based on the automation of decision-making processes on a national scale, are helping to achieve these expectations while improving government and social communications for every member of society. Electronic government fundamentally alters the dispersed governance system, affecting all aspects of document management and processing. Blockchain technology is a leading tool for implementing this governance. Numerous nations are attempting to incorporate technology into various government areas. This thesis will investigate the use of blockchain technology in e-governance and will put it to the test through its integration. Thus, it concludes that technology is still in its infancy, with plenty of room for improvement.

Key Words: Blockchain, Cryptocurrency, Decentralization, e-Government, Cryptography, Smart Contracts, Ledgers, Transaction, Encryption, Mining

Abstrakt

V současné informační kultuře se jednoduché, pohodlné a efektivní zapojení mezi vládou a občany stalo běžným očekáváním. Elektronická řešení státní správy, která jsou založena na automatizaci rozhodovacích procesů v národním měřítku, pomáhají naplňovat tato očekávání a zároveň zlepšují vládní a sociální komunikaci pro každého člena společnosti. Elektronická správa zásadně mění systém rozptýlené správy a ovlivňuje všechny aspekty správy a zpracování dokumentů. Technologie blockchain je předním nástrojem pro implementaci tohoto řízení. Mnoho zemí se pokouší začlenit technologii do různých vládních oblastí. Tato práce bude zkoumat využití technologie blockchain v e-governance a otestovat ji prostřednictvím její integrace. Dochází tedy k závěru, že technologie je stále v plenkách a je zde spousta prostoru pro zlepšení.

Klíčová slova: Blockchain, kryptoměna, decentralizace, e-Government, kryptografie, chytré smlouvy, hlavní knihy, transakce, šifrování, těžba

Table of Contents

1. Introduction to Blockchain Technology

For the aim of understanding, however, Blockchain works associate degreed what its capabilities are. It's essential to linger over a number of the underlying principles and technologies it was designed upon. The notion of blockchain first emerged in an exceedingly paper written by Satoshi Nakamoto in 2008. It introduces the digital currency Bitcoin, which depends on a decentralized and trustless platform, removing a central authority's necessity to validate transactions (1). It is supported three established technologies, a musical organization in a new way, public-key cryptography, a distributed peer-to-peer network, and a protocol implementing an incentivization system and record keeping (2). Whereas initially suffering from slow adoption from each world and business, Bitcoin itself, similarly because the conception of cryptocurrencies and Blockchain, typically fleetly gained traction. To date, there are around 1750 different cryptocurrencies listed on over two hundred exchange platforms, presently having an additive capitalization of over $ 263 billion, with a record high of virtually $ 800 billion in Dec 2017 (3), as displayed in Figure 1 (orange denotes the Bitcoin market capitalization, grey represents all different cryptocurrencies' market capitalization combined).

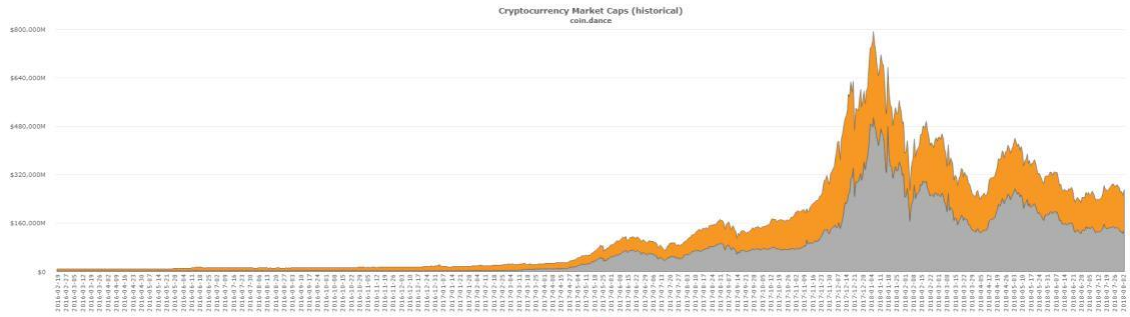


Figure 1 - Cryptocurrencies Market Cap (4)

This section can start with presenting details on the definition and cryptological foundations of the technology. It'll proceed to elaborate on the most characteristics, structure, and processes concerned in its operations. The present sorts and governance models will be examined in additional detail.

2. Objectives and methodology

2.1. Objectives

This thesis aims to delve into the implementation of blockchain technology in e-governance and test it out in its integration. During which will occur some problems and what solutions it might give away during its implementation. Several countries are trying to integrate the technology in different sectors of the government. This paper will thus focus on the functionality and integrity of the technology.

2.2. Methodology

The methodology of this bachelor's thesis is based on the study and gathering of information on the topic from the literature, the analysis of professional information sources and case studies, and the comparative method of applications of the technology. It will put emphasis on the architecture, application, and security of the Blockchain network.

It will then head on to use several platforms mentioned in the paper to test out the Functional aspect of the blockchain and test out the data integrity in cloud-based platforms and statistical explanations and discussions.

Subsequently, it will put conclusions justifying the objectives of this thesis with the help of findings in the experimental part of the paper.

3. Literature Review

3.1. Definition of Blockchain

Although the underlying blockchain technology was conceptualized and revealed in Nakamoto's written report in 2008 (1), a widely accepted definition wasn't a part of this paper and has not been developed till today. Jeffries, in her article" 'Blockchain,' is meaningless," (5) collects impressions from varied fields and use cases, reminiscent of governments, finance, law, etc., and concludes that" establishing a clear definition can facilitate clear up some (of these) misunderstandings." Walch (6) notes that the unclearness of the vocabulary used relating to the technology is thanks to many factors reminiscent of" word taint," the development of some terms getting undesirable meanings; technology variation, the continual development of all underlying technology that Blockchain depends on; and" crossfield communication," the fact that periods are utilized across multiple fields and industries with variable degrees of technological expertise. The International Standards Organization is presently engaged in the quality ISO/TC 307, which aims at shaping and standardizing varied aspects of Blockchain and distributed ledger technology such as terminology, security, privacy, and interactions through smart contracts (7). A definition of Blockchain devised by Seebacher and Schultz (8), additionally to be adopted during this paper, states the following:

A blockchain may be a distributed database that is shared among and specified by a peer-to-peer network. It consists of a connected sequence of blocks, holding timestamped transactions secured by public-key cryptography and verified by the network community.

Data is kept on the Blockchain, and it can be distributed across all nodes on the network, thus, forming a decentralized network (9). All collaborating nodes agree on a collection of rules before changing the blockchain network's integrity and operating accordingly. Counting on the specifics of the rules, the network's type, aim, and practicality is determined. One of the fundamental characteristics of a blockchain network is its immutability, i.e., all records created are permanent and may be modified solely at a high cost. Many styles of blockchain networks exist. They're

typically characterized supported the agreement and governance models employed, namely, decentralized (public, not permission), hybrid (consortium, permissioned), centralized (private/ ruled by one organization, permissioned). (10) (11)

3.2. Cryptography

As critical different strategies wont to hide information, cryptography aims to form transmitted messages incomprehensible to outsiders (12). The most data security objectives it pursues are the following:

- Confidentiality: reassuring that content is merely obtainable to those licensed to possess access to its information
- integrity: preventing unauthorized data manipulation
- Authentication: includes each entities authentication similarly to information and data identification; some reliable mechanism is a gift that assures that entities are who they claim to be which data has not been changed by unauthorized parties
- Nonrepudiation: preventing the denial of previous actions or transactions.

These characteristics are essential for Blockchain and form one among its most valuable components. Achieving confidentiality once transferring data over insecure channels is completed by employing an associate degree encryption scheme. Whereas there are several different approaches to encryption, three usually used strategies stand out: atonal encryption, rhombohedral-key encryption, and asymmetric-key encryption (also brought up as public-key). Atonal encryption cryptological tools (primitives) are discretionary length hash functions, unidirectional permutations, and random sequences. Some symmetric-key primitives are, for example, signatures, symmetric vital ciphers, identification primitives, etcetera Public-key ciphers, signatures, and identification primitives are samples of public-key primitives. Key-based encryption is predicated on the institution of a critical try only notable to the users who must be compelled to exchange information secretly. It’s attainable for one person to transmit a message that will solely be recovered by the one who has the second key. In symmetric-key encryption, the key for encryption and decipherment is the same, whereas in asymmetric, a try of different keys is

necessary. Key encryption is in situ to permit similar reusable information transformations to occur repetitively.

In contrast, the keys are often modified if suspicion that an unauthorized party obtained access occurs or as an alternative as a precaution. The safety cryptography provides supported the idea that the other information (e.g., the algorithmic specification), excluding the actual essential try in use, is or is public knowledge. Whereas keeping the particular encryption and decipherment transformation secret may in theory offer extra security, in practice, it's exhausting to assure those are unbroken secret—nevertheless, the secrecy of solely the key pair suffices to ensure a sufficient level of data security.

3.2.1. Unkeyed Encryption

Cryptological hash functions are a usually used tool for reassuring data integrity and message authentication. As a result of these characteristics, they're additionally one of the building blocks of the Blockchain. They take an input (message) and turn out a hash price or just a hash as an output." The basic plan of cryptological hash performs is that a hash value is a compact representative image of an input string and may be used as if it were unambiguously placeable in addition to that string." Typical usage of an atonal hash function would proceed with the subsequent steps:

- The hash value of a message is computed then is kept safely.
- At a later purpose in time, another hash of the first message is calculated.

A hash function has two essential properties: Compression:

- the process takes associate degree input of discretionary length and produces an output of a set bit size
- simple computation: the hash is simple to figure, notwithstanding the first input

Reflective on more specific properties, the modification detection codes (MDCs) atonal hash class is of interest within the Blockchain. The primary of those properties is preimage resistance

(also machine noninvertible), implying that thanks to the intense level of computational complexity, it's impossible to guess or generate the input supported by the ensuing hash output. The second property is collision resistance; a collision signifies the event of a hash performs manufacturing a similar production for two different inputs; applying MDCs, it's computationally impracticable to succeed in such a collision. Thus, it is accepted that the input is that the same if the hash is the same; labeling the hash function is deterministic.

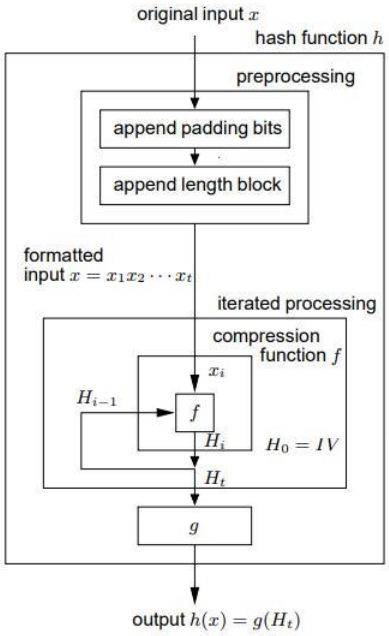


Figure 2: Iterated hash function algorithm (25)

As illustrated in Figure 2, a hash function depends on repetitive processes that take associate degree input of indefinite length then format it into separate fastened length blocks. The subsequent steps are typically followed:

- The hash input x is split into blocks of fixed length xi, and additional information, reminiscent of the length of the first input and extra bits to succeed in a definite bit length, is appended.
- Then, each block xi acts as associate degree input for the compression performance of the hash function h, which ends up in an intermediate result of fastened length.

The steps higher than are continual till the ultimate output reaches a predefined length. A final elective transformation g is employed as a final step, the so-called identity mapping. Relating to

the safety of hash functions, ideal security is considered a gift if manufacturing a preimage (the initial input) of a hash function needs 2^n operations, and simulating a collision needs $2^{n/2}$ processes.

Within the context of Blockchain, an actual application of hashes is that questionable hash purposes." A hash pointer may be a pointer to wherever data is kept at the side of a cryptological soup of the worth of that data at some fastened point in time." additionally to providing the suggests that to retrieve data, a hash pointer offers the likelihood to verify that the stored information has not been changed since its initial storage.

3.2.2. Symmetric-key Encryption

Similar secrets won't characterize Symmetric-key encoding to encrypt and decode the information transmitted between parties over an associate degree insecure channel. Two categories of functions are usually employed in practice: block and stream ciphers. Generally, it's assumed that the sole data that has got to be an unbroken secret once victimization key-based encryption is that the key itself. within the case of symmetric-key encryption, meaning that each party must assure that their keys (also referred to as secret-key ciphers) are kept secret as they're identical, and revealing any would jeopardize both the confidentiality and also the authentication of the communication. The most crucial challenge for symmetric-key encryption is the questionable fundamental distribution problem. To assure the integrity of the vital exchange transaction, it's sometimes necessary to encode the key in an exceedingly different key, implying the recipient already possesses that key. Despite this challenge, the most advantages of victimization symmetric-key cryptography lie the following aspects: the power to make ciphers with high rates of information throughput; the comparatively short, nevertheless cheap, keys used which offer a high level of protection; algorithms that are compared to others inexpensive to process; the fact that ciphers are wont to create stronger ciphers through easy transformations. (13)

3.2.3. Public-key Encryption

Public-key cryptography, additionally brought up as asymmetric-key cryptography, is predicated on the existence of a try of keys, namely, a non-public key associate degreed a public key, every appointed to an entity that has to verify its identity electronically or to encode information. The method is based on the premise that the general public secret's created overtly obtainable, whereas the private key is an unbroken secret. Messages that are encrypted victimization the public key will solely be disclosed through the employment of the corresponding private key. The most advantages of using public-key cryptography are the following:

- The transmission of encrypted data over insecure channels with the receiver's power to decode the information upon receiving it.
- Nonrepudiation: the sender will not deny the fact of causing the data or the data being altered at associate degree purpose throughout the transmission.

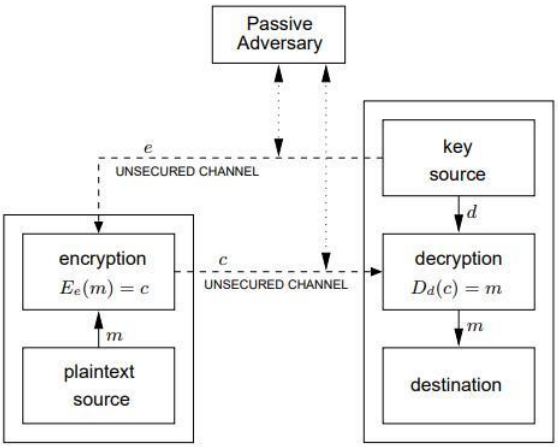


Figure 3: Public key encryption (25)

Figure 3 illustrates the process of public-key encryption between two parties:

- ✓ The entity B establishes a key pair consisting of the private key *d* and the public key *e*.
- ✓ B sends the public key *e* to the lefthand party A over an unsecured channel but keeps *d* private.

- ✓ A can use an encryption transformation to encrypt and transport a message to B using the public key *e*.
- ✓ The message may then be decrypted by B using the private key *d*.

Compared to symmetric-key encoding, public-key encryption needs additional calculations, making it inappropriate for enormous amounts of data. However, a possible, usually applied workaround is to use public-key encryption to send a symmetric key, thus, finding the critical distribution drawback. Diffie and playwright conceptualized uneven key encryption to unravel the urgent distribution problem within the first place. Furthermore, the public key is transferred over an insecure channel (as evident in Figure 3), enabling anyone possessing the general public key to send encrypted messages that solely the essential private entity will decrypt. (14)

3.2.4. Signatures

Aboard the encryption as mentioned earlier methods, digital signatures have applications for authentication, conserving information integrity, and providing nonrepudiation." A digital signature of a message may be a range obsessed with some secret notable only to the signer, and, additionally, on the content of the statement being signed." this enables the message to be digitally connected with its origin. one among the essential uses of such signatures is for the verification of public vital owners' identities and, as a consequence, preventing that the sender of a message denies the dealings at a later point. Functionally, a non-public key assumes the role of a digital signature, thus, authenticating the origin of a transmitted message. The message receiver will then utilize the general public key to decode the message, therefore, establishing the sender's identity. If a sender denies authoring a statement, the match between their private key and the public key wont to decrypt the message is Proof that they were the first author.

3.3. Characteristics of Blockchain

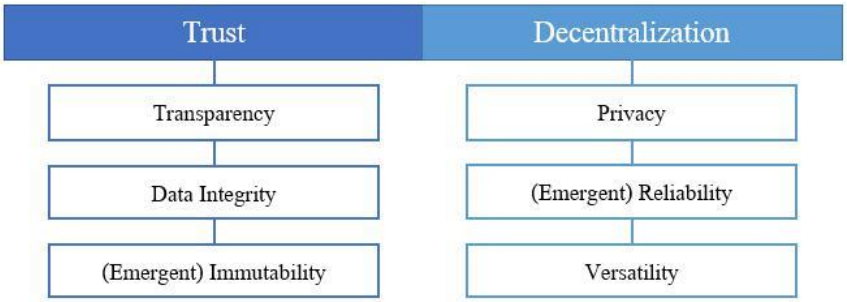


Figure 4: Blockchain characteristics (8)

In keeping with Seebacher and Schuritz's literature review, blockchain technology creates a setting that will be best characterized as decentralized and trust inspiring. Thanks to the system's design and the technologies it was designed on, as illustrated in Figure 4, more parts form these facets. Transparency, information integrity, and unchangeableness underline the trust perspective, whereas privacy, reliability, and flexibility strengthen the worth decentralization delivers. More details on every aspect are going to be provided during this section.

3.3.1. Trust

Blockchain's provision of transparency is alleged to support the peer-to-peer nature of the network it's built on, enabling shared and public interactions among the participants. Thanks to the fact that every transaction is publicly seeable and broadcasted to the whole network, all actors In the system can acquire timely and thorough information. Furthermore, since a central authority who may singlehandedly influence transactions, permit them, discard them, or alter them is replaced with a trust model supported networkwide consensus, transparency and trust are essential characteristics of Blockchain.

The technology additionally facilitates information integrity due to the reliance on public-key encryption and hashing that modify high-security levels and govern the direct interaction with data keep on the system.

The third necessary characteristic of blockchain causative to declarative trust is its questionable unchangeableness. Theoretically, once a dealing is approved by the network participants, further to a block, and the league is added to the Blockchain, it can't be modified. This transformation is processed through the employment of agreement mechanisms, reminiscent of Proof of labor and Proof of stake, that permit the participants to demonstrate their trustworthiness. Whereas the immutability claim is often cited, it should be noted that a" sufficient quantity of computing power" is wont to alter a blockchain (for example, within the case of the Ethereum decentralized Autonomous Organization fork in 2016). Thus, Conte Diamond State Leon et al. describe the unchangeableness property of Blockchain as" emergent," that's to imply that it's not incontestible nevertheless is desired.

Additionally, it is a challenge to say with certainty that such characteristics hold in reality. Some analysis (e.g.) has been done to analyze blockchains' performance and security, specifically the Proof of labor algorithm. In contrast, others reminiscent of Cucurull and Puiggali created a proposal that" enhances the safety of the immutable logs" and provides more integrity and nonrepudiation by" publishing log identity proofs on the Blockchain."

3.3.2. Decentralization

Within Blockchain, trust and decentralization are interconnected as transparency, information integrity, and unchangeableness is necessary for creating a decentralized network. Decentralization, on the other hand, together with the aspects of privacy, reliability, and versatility, is important to assure the participants' participation. To elaborate, privacy in an exceedingly blockchain-based system is builtin through the employment of public and personal keys that manifest users while not the necessity for revealing real identities. whereas transactions are derived back to a particular key, thus, creating the network pseudonymous, privacy continues to be enabled. , Blockchain's dependableness is an associate degree other facets that are tagged as" emergent" thanks to bound risks that, whereas unlikely and machine difficult, still exist. The reliability claim is predicated on two factors: firstly, the data and transactions being shared and kept in an exceedingly distributed manner across the network, and secondly, on the machine-driven nature of the system. However, Conte Diamond State Leon et al. argue that if the Blockchain has

been tampered with, for example, through an individual usurping fifty-one or additional p.c of the computational power within the network, such an attack is left unnoticed, so compromising the reliability of future transactions. Suppose one party controls a minimum of 51% of a blockchain network. In that case, they will stop new transactions from gaining confirmation through the agreement protocol, so they block different users from victimization the network as intended. Relating to versatility, the character of blockchain technology permits developers to contribute their own code and extend the appliance landscape, thus enabling the event of a range of use cases and more shifting aloof from centralized systems.

To better comprehend blockchain, let us examine how Bitcoin implemented it. Bitcoin, like a database, requires a network of computers to keep its blockchain. In the case of Bitcoin, this blockchain is just a special form of database that records every Bitcoin transaction ever made. However, in Bitcoin's databases, these computers are not all housed in the same building, and each computer or set of computers is run by a distinct individual or group of individuals.

Bitcoin is made up of thousands of computers, yet each computer or set of computers that stores its blockchain is located in a different geographic location and is managed by a different individual or group of individuals. The machines that comprise Bitcoin's network are referred to as nodes.

Each node in a blockchain maintains a complete record of all data placed on the blockchain since its beginning. The data for Bitcoin is the complete history of all Bitcoin transactions. As a result, if a node's data has a mistake, it may utilize the thousands of other nodes as a reference point to repair the issue. In this manner, no node in the network may modify its data. As a result, the history of transactions within each block that constitutes the Bitcoin blockchain is irreversible.

Consider what would happen if one user tampered with Bitcoin's transaction record; all other nodes would cross-reference and readily find the node that contained the wrong information. This system contributes to establishing a precise and visible sequence of occurrences. This information is in the form of a list for Bitcoin.

3.4. Blockchain Structure

3.4.1. Blockchain

The Blockchain as a concept, i.e., not concerning any specific application however rather to the information structure overall, maybe a combination of blocks. As a consecutive dealings database, it keeps a record of all transactions that have occurred, bundled in interconnected timestamped blocks (Figure 5). the primary block in every Blockchain is named a genesis block and is outlined by the developers of the particular Blockchain. Thanks to the interlacing of all blocks supported hashes. Any changes are created within the Blockchain when the acceptance of a given block is derived back as all blocks have a similar genesis block. Every new block contains the hash of the previous block. Therefore, if something is altered, the hashes won't match, so demonstrating inconsistency and tampering.

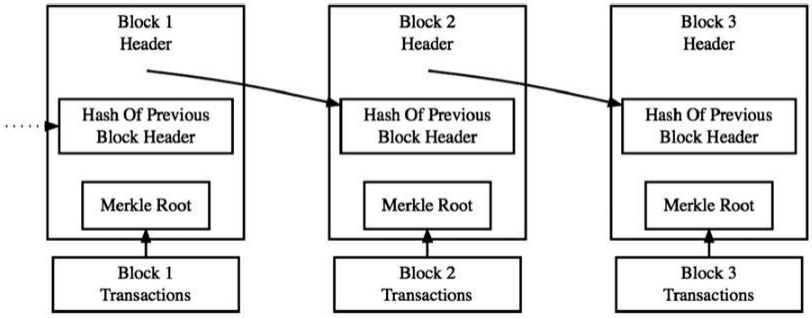


Figure 5: Simplified blockchain structure (26)

3.4.2. Block

As evident from Figure 5, every block within the chain contains two objects: a block header containing data and a bundle of dealings. The block's header helps to verify the validity of the transactions keeps in it. It generally contains the subsequent information: the present version of the block; the header hash of the previous block; Merkle root hash (hash of all transactions enclosed in the block); a timestamp; a present (random price miners modify to unravel the machine puzzle). Typically, each transaction that's new to the network is enclosed within the latest block.

Generally, blocks were introduced into the blockchain network to optimize its operations, and efficiency, i.e., rather than corroborative individual transactions on the network, nodes will handle bundles of multiple transactions forming a block simultaneously. Block size is one dimension that influences the operations of the network because it indirectly defines the number of transactions included in an exceeding block, that influences the general output of the system, i.e., larger blocks lead to lower propagation speed, which ends up in additional questionable stale blocks, resulting in a less secure network. Stale blocks are specified; they are not enclosed within the longest chain, thanks to conflicts. They will trigger chain forks, thus, speed down the general growth of the Blockchain and jeopardizing its performance and security by giving a plus to potential attacks, reminiscent of double-spending.

A Merkle tree, named when its creator Ralph Merkle, may be an organization that represents a binary tree designed victimization Hash pointers. All blocks containing information are sorted into pairs of 2, and also the hash of every one of them is kept in an exceeding parent node. In turn, the parent nodes are grouped into pairs of two and stored in another parent node one level up the tree. This procedure is continued till all blocks are sorted into one block that's the foundation of the tree. The foundation of the Merkle tree is kept within the block header and helps assure the integrity of the block and also the dealings enclosed in it.

3.4.3. Transaction

A transaction is that the smallest building block of a blockchain. It consists of a sender address, a recipient address, and a value. There are bound rules that every transaction has to obey. one among these states is that for every transaction on the Blockchain, the input must equal output, wherever these represent any tangible or intangible value. Extra rules rely upon the particular blockchain application and are sometimes enforced through the employment of a scripting language reminiscent of Forth in Bitcoin. Such rules are the underlying infrastructure for smart contracts, which can be represented intimately within the following section.

Any dealings on the Blockchain basically modifies the state of the whole chain. Thanks to the shared, decentralized, and distributed nature of the ledger, all nodes on the Blockchain that

severally hold a replica of the chain have to be compelled to method the transactions to get an updated copy. As noted previously, dealings are bundled in blocks associate with degreed unfold throughout the network. Afterward, they're checked for correctness and validity by every node following the predefined rules. Every transaction includes a minimum of one input and at least one output. The additive value of all inputs should equal that of all outputs to assure that no partly unexpended transactions occur. Thus, the worth that isn't transferred to a different entity is additionally tagged as an output (unspent transaction). The sole difference between input and output allowed, wherever input is larger, is to account for transaction fees that are paid to the laborer who establishes the actual block.

Additionally, output values can't be larger than input values to forestall the creation of inexistent values on the Blockchain, reminiscent of cryptocurrency. Associate degree input may be with respect to an output from a previous transaction that permits for a verifiable stream of the transferred value. The output consists of a quantity and an address. In addition, every input additionally includes a digital public signature/key of the sender, scriptSig, signifying their credibleness and certifying that they're after all allowed to send the enclosed value. Solely the entity in possession of the matching personal key has access to defrayal the dealings output. (2)

3.5. Network Operations

The subsequent subsections are dedicated to elaborating on the {ways|ways that|ways in that} blockchain networks operate, specifically, how their integrity is ensured by different agreement mechanisms and the way new data is further to the chain. Whereas many styles of blockchains exist, namely, decentralized, permissioned (centralized), and hybrid, the processes below specialize in the classical decentralized perspective.

3.5.1. Consensus

A consensus mechanism is a vital facet of a blockchain-based system, and it's tightly interlinked with its peer-to-peer distributed nature. The implementation of distributed consensus in an exceeding network with multiple nodes, a number of which are potentially malicious, aims at establishing dependableness associate degreed includes two aspects: firstly, delivering a price (in this context, a block) on that all honest nodes agree and secondly, reassuring that the worth was generated by an honest node. Thus, the formula permits the community as a full to verify that every block projected by a node is legitimate and, in an exceeding addition, serves to avoid conflicts. So as to try to do this, all nodes on a blockchain abide by a collection of reciprocally specified and self-enforced rules that are outlined, counting on the nature, purpose, and underlying assets in a blockchain. Therefore, the additional nodes participate in an exceeding network, the stronger the agreement mechanism becomes, thanks to the number of actors enacting their own rules. The fundamental rules a blockchain follows are the following: 1) the valid chain is that the one that contains the genesis block; 2) it contains solely blocks deemed valid in keeping with the chain's rules; 3) the chain is the longest that exists within the network, i.e., together with the foremost blocks. Furthermore, a consensus mechanism will vary in style, counting on the particular cryptological validation technique encoded in the distributed ledger. The Bitcoin ledger, for example, utilizes the" proof of work" mechanism to produce agreement {in a|during a|in associate degree exceedingly|in a very} globally distributed blockchain network. More details on" Proof of work" and an alternative, namely," Proof of stake," are bestowed within the following sections. The final consensus reaching method entails the following: at regular intervals, for example, every 10 minutes, one haphazardly selected node broadcasts a group of nevertheless unapproved transactions for an ensuing block on the chain; then, every node executes a consensus protocol, checking the validity of the projected transactions; if the consensus protocol is run successfully, a legitimate block is going to be selected, and its hash will be enclosed in the next block. Although some valid transactions weren't enclosed within the projected block, they're going to stay outstanding and expect the ensuing valid block they will be proposed and included in.

Pseudonymity, as antecedently mentioned, maybe a goal pursued by some blockchains, reminiscent of Bitcoin. Whereas some transactions can still be connected to 1 another, no node is forced to reveal its true identity. Additionally, as a result of the shortage of authority dominant the

identities of the network participants, some nodes can produce copies, all controlled by one individual with the aim to maliciously influence the constructions of blocks, so getting a better degree of control. This is often brought up as a Sybil attack. The shortage of identities and also the simple making of extra nodes produce difficulties for the safety of the agreement protocols. This challenge, involving the Byzantine Generals Problem, common in distributed systems, is overcome by choice of a random node through the employment of a mining puzzle. That node obtains the correct to propose ensuing block to be enclosed within the chain. Just in case that node is detected to be malicious by different nodes, it'll not be verified, thus, will not be enclosed within the chain, therefore, implicitly reaching an agreement that the node isn't trustworthy and lengthening the chain with associate degree other valid blocks. Nevertheless, there are risks that attack reminiscent of the double-spend attack, in the case of a cryptocurrency-based blockchain, occur in the network. Such an attack is characterized by a particular node trying to pay a similar coin doubly by 1st causing a coin to a different user then provision dealings that returns the at the start spent coin back to an address controlled by the payer. Therein case, if the second transaction is valid and included in an exceedingly block, the coin is spent twice. However, thanks to heuristic mechanisms reminiscent of nodes corroboratory the blocks that they 1st detected within the network and, additional importantly, incentives, the probability of such attacks are drastically reduced.

3.5.2. Mining Puzzle

The method of making a replacement block and also the real-world resources invested within it are altogether brought up as mining, a vital component of all blockchain-based systems. From a technical perspective, mining involves the steps printed above, namely, validating, storing, and building a dealings tree of all transactions declared in the network in an amount of time. In order to construct and propose a block out of this information, at the side of the attendant metadata, a node should notice the answer to a mathematical puzzle. Once the solution is found, it's additionally enclosed within the block, and it is then broadcasted to the network. Then, the network node can use the agreement mechanism explained higher than to validate the block and domestically store it. The mining puzzle not solely assures the randomness of the node that proposes a block however also regulates the time intervals at that new blocks are issued. For

example, in the Bitcoin network, the typical time it takes for a replacement block proposal is 10 minutes. Thanks to the interdependence between the provision of resources (computing power) and also the time it takes to unravel a puzzle, the puzzle difficulty is adjusted to assure the approximate regularity of the interval. The introduction of such a mining puzzle prevents malicious nodes from posting faux transactions and threatening the integrity of the network, thus, acting as an answer to the Byzantine Generals' Problem. whereas multiple mathematical puzzle formats exist and are applied across different blockchain networks, the foremost widespread ones are" Proof of work" and" Proof of stake," which are explained below.

3.5.3. Proof of Work

The agreement mechanism is that the most generally used consensus formula among existing blockchains. The underlying plan is that the choice of the random node that forms the ensuing block within the chain is proportionately supported by the resources obtainable thereto node. Within the case of Proof of work, the resource in question is computing power. Supported the notion that a minimum of fifty-one percent of the network should be controlled by one entity for a Sybil attack to be launched, thus, nullifying the Blockchain's integrity, the assumption is that it's getting ready to not possible for associate degree individuals to succeed in such a high level of management over the computing power obtainable {in a|during a|in an exceedingly|in a terribly} blockchain network. The operation of the POW formula is going to be detailed on supported the Proof of labor system employed in Bitcoin.

The Bitcoin network obtains proof of work victimization hash-based puzzles. So as to propose a replacement block, a node has to notice a random price (nonce) that concatenated with the hash of the block falls in a bound target value bracket. This target area may be a very tiny set of the general output space. Forward that hash functions are cryptographically secure, making an attempt all attainable mixtures is that the sole thanks to noticing an answer to such a problem. Therefore, it's doubtless that the node which can find the answer to the matter is the one that has the foremost computing power available. Hash puzzles have three necessary properties. The primary one is the machine difficulty, which will increase over time, each 2016 block, to adapt to the increasing handiness of computing power within the network. The second property ensuing

from this adjustment is the variability of the price for a block. Considering the increasing difficulty and also the continuous growth of the network, the effort to mine a block will increase over time, thus, either increasing the demand for additional computing power, i.e., hardware, or decreasing a node's possibilities to seek out a block. The third essential property is that the simple corroborative answer to the puzzle revealed by a node. On average, it takes a node 1020 to do to find a present that fulfills the condition that the hash falls below the target, which nonce should be published as a part of the block. Once that's done, the other node will verify that the hash of the block's contents is below the target value. This asserts the decentralization characteristic of Blockchain as no central authority is required to verify the honestness of the blocks. The hash perform employed in Bitcoin is double SHA256, which ends up in an exceedingly 256bit (64character) output. The concrete drawback is to seek out a block whose hash starts with a given number of zero bits, wherever the amount denotes the difficulty of the puzzle. Whereas Proof of labor may be a reliable mechanism for reassuring the integrity of the network, there Are serious concerns over the intensive power usage and questionable property of the present design. Diamond State Vries estimates that as of March 2022," about twenty-six large integer hashing operations are performed each second" within the Bitcoin network, whereas it processes solely twenty-three transactions per second. Whereas hard the whole power consumption for the Bitcoin network remains a challenge thanks to its distributed and close nature, in keeping with a report revealed by the International Energy Agency already in 2022, it surpassed the yearly usage rates of Thailand (195 TWh), among others, and is projected to grow to one hundred twenty-five TWh each year by March 2022 (Figure 6) (15). additionally to the energy utilization concern, Bitcoin mining is dominated by specialized hardware creating the method loads additional efficient, namely, associate degree application-specific microcircuit (ASIC), that is tailor-made for a particular use, during this case, the mining of Bitcoin, and outperforms any graphics process unit (GPU), effectively pushing the miners counting on those out of the mining market. This tendency threatens the decentralization conception of the Bitcoin network, move a challenge for one among the basic characteristics of Blockchain.

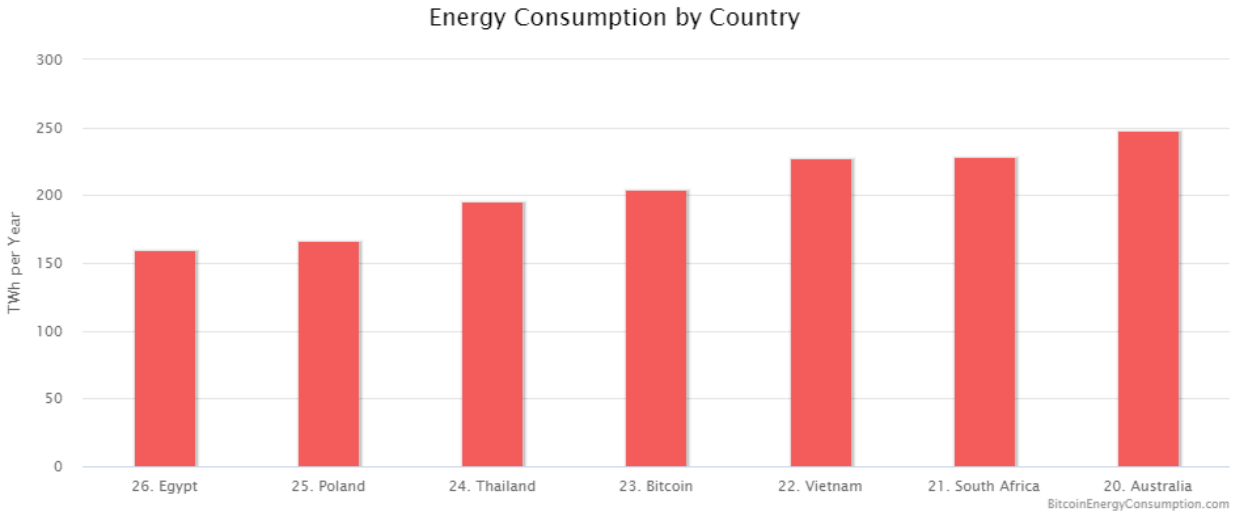


Figure 6: Energy Consumption by Country (15)

Incentives: Proof of labor and specifically Bitcoin (and different cryptocurrencies) believe a model with two separate rewards to incentivize honest behavior in laborers. The primary one is that the block reward. Every recently created block includes extra coin creation dealings that grant the miner the correct to distribute the reward by either causing it to associate degree address of their own or elsewhere. By provision the reward, the node is incentivized to behave honestly so as to possess different nodes further extend the agreement chain. The block reward is fastened for the period of mining 210,000 blocks, which commonly takes around four years. Currently, the reward is 12.5 Bitcoin, whereas, at the origin of the currency, it was 50. This mechanism is enforced to make sure the finite amount of obtainable coins, twenty-one million, mimicking real-world currencies. The second incentive mechanism in Bitcoin may be a dealings fee. The difference between the input associate degreed output values for any transaction is reserved by the creator of a transaction and issued to a block laborer as a fee for the transaction being enclosed therein block. Whereas such action is voluntary, additional nodes follow the system so as to assure a decent service level on the network.

3.5.4. Proof of Stake

Proof of stake (PoS) is another agreement protocol designed for public blockchains. It absolutely was designed as an alternate to Proof of labor to counteract, among different issues, the energy consumption issues bestowed previously. Whereas several PoS variants are developed, the mechanism isn't as widely adopted as PoW, with the foremost outstanding blockchain victimization it's Ethereum. The nodes in PoS are denoted as validators, and rather than mining the Blockchain, they validate dealings so as to earn a transaction fee. The ability of every validator is predicated on the proportion of coins command by the node. Therefore, once the agreement formula haphazardly selects a node to propose the ensuing block and broadcast it to the network, the likelihood that a node is chosen depends on the quantity of stake it holds. So as to become a validator, one will send special dealings that turn the cryptocurrency they hold into a deposit, i.e., their stake.

The most advantages of PoS over POW embody the subsequent aspects :

- the safety of the Blockchain isn't obsessed with giant energy consumption. Thus, the network is additional property within the long run.
- There are additional opportunities to employ techniques that discourage the formation of centralized unions and thus jeopardize the decentralized nature of Blockchain.
- Possessing a larger stake not only entitles you to proportionate gains as a critical PoW, but also provides you with the means to acquire additional computing power.
- Ability to introduce economic penalties for 51% attacks, thus, creating them loads dearer as security within the network originates from putt price at a loss thanks to the validators' deposit.

Despite these positive aspects of the PoS protocol, there are many vulnerabilities that need more development to assure the broader adoption of PoS. Most of the problems are due to the actual fact that PoS isn't tuned in to any external factors that occur on the far side of the Blockchain (in opposition, machine power in POW is an associate degree external factor); therefore, there's no affiliation between PoS and also the physical world. The primary drawback is that the questionable" nothing at stake" concerning the power of a validator to, at the same time, mint

blocks on multiple branches/forks once they occur accidentally or deliberately. Such a scenario will build it easier to double pay or issue different attacks on the Blockchain. Another potential threat is the" long-range attack" at an assaulter with a sufficient stake who can decide to build an alternate chain beginning either at a random block or even at the genesis block. more weaknesses embody the likelihood for double-spending, the initial distribution problem, which provides a plus to entities that joined the chain at associate degree earlier purpose in time, etcetera Furthermore, the social element of PoS and also the inability to objectively confirm the PoS system state primarily based solely on protocol rules weaken its" decentralization facet and mathematical soundness." Modification of PoS similarly as hybrid PoWPoS models, reminiscent of Slasher, are developed and enforced in varied blockchain systems in a shot to mitigate the abovementioned risks. Delegated PoS systems offer the validators the chance to 1st vote on a gaggle of delegates who then have the right to come up with blocks one when the opposite in exchange for a bequest and acquire disciplined just in case of malicious activity. During this setup, a block is formed by one user. However, it has to be signed by over one delegate. (16)

3.6. Ledger Types and Governance Models

When the initial blockchains reminiscent of Bitcoin and other cryptocurrencies gained popularity, more use cases and architectures for the dealings important in peer-to-peer networks were designed and implemented. The three styles of blockchains, counting on the amount of decentralization they are characterized by, are public (decentralized), hybrid, and personal (centralized). Thanks to the inherent decentralization facet of blockchain technology and also the claim that the private kind doesn't fulfill this aspect, the overarching" distributed ledger technology (DLT)" term is employed to consult with the three variations. Figure 7 classifies the kinds of blockchains in keeping with two features: obscurity and trust in an exceeding validator. Obscurity denotes the degree to those participants within the Blockchain are identified. Trust in a validator on the coordinate axis describes the degree to which penalization for wrongful conduct is inevitable; i.e., in a permissioned blockchain, nodes would like to be allowed to participate and acquire express rights to conduct bound actions Table one is customized from Kruijff and Weigand And summarizes the

most characteristics of the three styles of blockchains and provides a summary of the nomenclature wont to describe a similar notion across the literature.

Public blockchains reminiscent of Bitcoin have supported the Proof of labor agreement mechanism and that they aren't permissioned. Thus, anyone will participate in them and validate transactions. The character of the network additionally guarantees the pseudonymity of all nodes. Thanks to the open and unconditional participation character of the network, the level of trust among participants are outlined as" low."

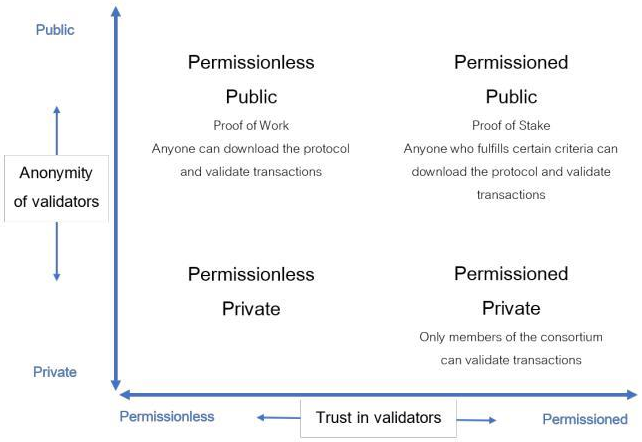


Figure 8: Types of Blockchains Trust vs. Anonymity (27)

Thanks to the antecedently mentioned giant resource demands, the measurability of those networks are limited. Ethereum, for example, is additionally a public blockchain; however, it's tagged as permissioned due to the usage of the Proof of Stake algorithm, i.e., one should purchase coins before having the ability to participate within the block validation process, thus, offering a better degree of trust in the validators in the network. Public blockchains have a high potential to disrupt multiple industries and varied business models, counting on intermediaries and trustworthy third parties to interact value.

Hybrid blockchains also brought up as association blockchains, are ruled by a preselected group. Such blockchains are quicker and additional scalable and offer a better degree of privacy for all dealings. The agreement mechanism is predicated on validation from a preselected set of nodes and a change of integrity. The network is additionally controlled by a closed group. Variable

degrees of reading rights are granted, wherever some chains may be receptive to the general public to view. Such a consortia setup offers a reduced rate of transaction prices and may replace a paper-based system to exchange documents among a closed cluster of stakeholders. Thanks to the hybrid nature, Kravchenko characterizes this sort of ledger as permissioned, private (Figure 7) as underneath bound social agreement anybody who qualifies will become a validator.

Private blockchains are under the management of one organization. Write permissions are monitored by that entity, whereas scan privileges could also be granted to a wider audience or restricted. A non-public blockchain may be a permissioned ledger, thus, enabling a high level of trust among the participants. Such blockchains are appropriate for applications reminiscent of auditing and database management that are internal to one company.

3.7. Application: Smart Contracts

As this paper can more linger over specific implementations of blockchain use cases, it's essential to look at the appliance layer of the system too. The foremost outstanding and notable wide application of Blockchain on the far side cryptocurrencies is predicated on smart contracts. The term originated from Nick Szabo's cryptography add 1996. It was outlined as" a set of promises, per digital form, together with protocols among which the parties perform on the opposite promises." Furthermore, concerning Bitcoingrounded smart contracts, technology applications used for mechanically transferring cash as a trigger event are widely A smart contract may be a mechanism that mechanically distributes digital assets between 2 or other parties, wherever some or all of them 1st deposit the assets into the smart contract.

Consensus	Type	Governance	Trust	Scalability
Decentralized Based on proof and cryptoeconomics	Public Not permissioned	Pseudonymous nodes/ No centralized management	Low	Limited
Hybrid Based on validation by a pre-selected set of nodes	Consortium Permissioned	Pre-selected set of nodes/ Multiple organizations	Medium	Unlimited
Centralized Based on validation	Private Permissioned	Single organization	High	Unlimited

Table 1: Public, Hybrid, and Private Blockchains

The main advantage of implementing a smart contract is that the execution is warranted underneath a specified set of conditions and is freelance of someone entity among or on the far side of the Blockchain while not the necessity for a central authority or a legal system. Additionally, smart contracts radically cut back dealings prices thanks to the employment of standardized rules expressed in digital type (machine-readable code). They will flip legal obligations into machine-driven processes, ensure a larger security level by being tamper-resistant, and reduce reliance on third parties reminiscent of intermediaries.

Bitcoin uses some smart contract typescripts to make sure the transfer of coins between parties. However, to modify the event of discretionary scripts, a Turing-complete programing language is needed. Ethereum accomplished this by building a separate" abstract foundation layer," thus, permitting anyone to make smart contracts with their own rules and formats (17). Whereas smart contracts offer virtually endless potentialities relating to what is enclosed in them, there are bound challenges to be thought of too. Firstly, a smart contract's mechanisms are somebody's creation, i.e., a programmer develops the code. Thus, it will still embody some logical errors that can't be accounted for by the contract itself. Secondly, while such agreements have an advantage thanks to their accuracy, cryptological and analytical safety, traceability, and unchangeableness features, they're not thought to be de jure enforceable contracts at the instant. They aren't accepted in an exceeding court of law. Finally, smart contracts solely believe data provided to them. Therefore, they can not take actions supported by external circumstances, which poses another challenge once considering the immutability aspect, i.e., there's no chance to change a smart contract kept on the Blockchain. (18)

Decentralized Autonomous Organizations (DAO) function a long extension to individual smart contracts of upper complexity, wherever organizations like structured are sculptural on the Blockchain. Rather than following ancient governance models, DAOs believe the distributed blockchain network and offer shares through Initial Coin Offerings (ICOs) to autonomous stakeholders, thus, enabling deciding and operations management through the employment of distributed agreement and smart contracts. The viability of such organizations and their attendant parts are still to be evaluated for their success and sustainability.

3.8. Regulatory Landscape

The continual development of varied applications of Blockchain, together with cryptocurrencies, ICOs, smart contracts, etc., in an exceeding multitude of industries, with a big dominance within the monetary sector, has raised several queries relating to the legal and restrictive implications of the technology. The quick pace of amendment causes the shortcoming of restrictive agencies worldwide to introduce relevant laws and regulations. The bulk of the discussions are fragmented and focus totally on ICOs and cryptocurrencies. As evident from BitLegal's map of regulatory attitudes toward blockchain technology, sentiments vary from permissive (mostly in North America and Europe) through" contentious" (in the majority of Asia) to" hostile" in states reminiscent of China (19) with loads of conditions remaining within the" unknown" state.

Twenty-three of the ECU trade unionist states signed a declaration on the institution of an ECU Blockchain Partnership on Gregorian calendar March 10, 2018, communicating a disposition to support the event of the technology for the advantage of the general public and private sector. However, the final information Protection Regulation (GDPR), effective since May 25, 2018, is thought to be a challenge for implementing blockchain applications involving personal data thanks to the clash between Blockchain's unchangeableness characteristics and" the right to be forgotten." Countries reminiscent of Japan and Germany acknowledge Bitcoin as a currency and have already obligatory taxes on it. In contrast, China, Indonesia, and South Korea have prohibited the employment of cryptocurrencies and ICOs. The G20 meeting in the Gregorian calendar month confirmed the member states' position on remaining alert relating to developments within the cryptocurrency space. It recognized the potential importance of technology as a full for" financial systems broader economy". From among the blockchain space, there's typically a positive perspective that" increased regulation, as long because it is completed with the cooperation of trade stakeholders and with the aim of de-risking the broader market, can hasten blockchain adoption by giant enterprise users and reassure institutional investors." Implications for legal and governance processes and structures and also a gift and are being acknowledged thus far principally in analysis with authors reminiscent of Raskin and Diamond State Filippi and Wright suggesting that smart contracts, specifically, and Blockchain generally have to be compelled to be

recognized for their potential and integrated among existing processes and controlled through" new rules and new approaches to legal thinking."

4. Practical Part

In this part I will use the information gathered from the literature review to formulate an experiment and achieve the objectives of this thesis. I will mainly focus on *functionality* testing and *data integrity* testing to fulfill my targets.

4.1. Functionality Test

4.1.1. Smart Contracts

Smart contracts are self-executing business automation applications that run on a decentralized network such as blockchain. In most cases, they are a set of digital contracts with a set of rules, and they can be enforced without the help of a third party. In the blockchain, many people work together to make smart contracts. Once the smart contract is deployed, it will be added to the blockchain network and sent to all nodes that can check it.

However, some researchers have said that writing secure smart contracts is not easy because of how the EVM works. A contract cannot be changed after being put into place, which means that any flaws could cost much money if exploited.

One of the most popular decentralized investment fund projects called DAO, was attacked in 2016. The figure below is a simplified version of the DAO contract used in the figure. It uses a mapping called balances to track how much money each project has raised. When a project gets enough support from the community, the owner can call the withdrawBalance function to get the ether out of the DAO. It was a mistake, but the transfer mechanism in DAO allowed the ether to be sent to an outside address before the internal state was changed. In this way, the attacker can steal more ether from the contract. Call.value would be called by the withdrawBalance function and the call.value method would be called by the withdrawBalance function by the caller's fallback function. Then there was another withdrawal. Because the balance array's value had not been changed, the contract would keep giving the attacker ether until the balance of the contract was 0, at which point it would stop. It cost $60 million and caused the hard fork of Ethereum, which caused many bad things.

```
contract BasicDAO {
  mapping (address => uint) public balances;

  // transfer the entire balance of caller
  function withdrawBalance() public {
    uint money = balances[msg.sender];
    bool result = msg.sender.call.value(money)();
    if (!result) { throw; }
    // update balance of withdrawer
    balances[msg.sender] = 0;
  }
}

contract Attacker {
  address public owner;

  // Initiates the balance withdrawal.
  function callWithdraw(address addr) public {
    BasicDAO(addr).withdrawBalance();
  }

  // Fallback function for this contract.
  function () public payable {
    callWithdraw(msg.sender);
  }
}
```

Figure 9: Algorithmic steps during transactions

4.1.1.1. Testing Technologies and Tools

Many people found flaws in smart contracts to ensure that everyone's money and privacy were safe. During a literature review, I looked at 27 papers published between 2016 and 2020 and used traditional software testing methods to look at the security of smart contracts. Table 3 shows them in order of when they were made. Here a quick summary will be given of the main ideas behind these tools, organized by the detection method they are primarily based on. When the conclusion comes, how they evaluate their proposed methods will be discussed, which will be discussed in more detail in the next section.

Static Analysis. Static analysis is a fashion for studying a program without executing it, and it may be used at both the source law and bytecode situations. While stationary analysis ways may examine an entire law base, they can induce numerous false cons. SolidityCheck parses each function at the source law position and compares it topre-defined vulnerability patterns. Generally, other tools will admit double bytecode and also use it to induce a bespoke intermediate representation, similar as SSA used by Slither, Datalog used by Security and MadMax, XML parsing tree used by SmartCheck, and XCFG used by Extrasensory perception. This format defines and matches vulnerability patterns to screen out questionable law particles. Formal verification, as a static analysis fashion grounded on fine attestations, is also generally used to insure the logical integrity of smart contracts similar as EtherTrust, VeriSmart, and Zeus. Also, stationary analysis may be employed to prize features that can be used to train classifiers.

Symbolic Execution. When symbolic execution is used to examine a program, symbolic values are used as input rather than execution. When a fork occurs, the analyzer gathers the appropriate path constraints and then uses a constraint solver to determine the particular values triggering each branch. While symbolic execution allows for simultaneous exploration of several pathways a program might follow in response to various inputs, it also introduces inherent difficulties such as path explosion. Typically, the symbolic executor will construct a control flow graph based on Ethereum bytecode, establish relevant constraints depending on the features of smart contract vulnerabilities, and then generate good test cases using the constraint solver. For instance, Oyente, Mythril, and its derivatives. There has been an ongoing effort to improve the symbolic execution process in recent years. For example, Manticore offers support for unconventional execution contexts, DefectChecker pulls defect-related characteristics to aid in performance optimization, sCompile detects essential routes involving monetary transactions, and VerX focuses on validating external callback-free contracts successfully.

Dynamic Fuzzing. Fuzzing is a technique for detecting software faults that involve creating unexpected input data and observing the target program's anomalous results during execution. It enables developers to assure a consistent level of quality through the use of pre-defined tests but does not provide insight into the reasons for reported defects. When used to smart contracts, a fuzzing engine will attempt to produce initial seeds before constructing executable transactions.

Then, it will dynamically alter the produced data in response to test outcomes to explore as much smart contract state space as feasible. Finally, it will assess each transaction's status using the finite state machine to determine whether an attackable threat exists. ContractFuzzer pioneered the use of fuzz testing for smart contracts. Later on, additional scholars investigate ways to improve various aspects of fuzzing. At the same time, ReGuard and Harvey focus on producing varied inputs and transactions that are more likely to expose vulnerabilities, ILF and sFuzz focus on developing a more effective generation of mutation strategies.

4.1.1.2. Related Work

Numerous empirical studies have focused on smart contract test suites and test techniques in recent years. Several detailed classic test approaches, concentrating on their inherent limitations, such as fuzzing's instability and symbolic execution's state explosion problem. Others focused on developing benchmarks for evaluation via web crawling, bug injection, and patch improvement. Additionally, they will run several vulnerability detection programs to undertake exploratory trials to establish the benchmarks' quality.

In contrast to the previous work, It would provide a systematic, complete analysis of the many aspects that might influence smart contract testing tools' assessments and suggest a recommended evaluation technique for future reviews of such tools. In contrast to empirical research on fuzz testing and symbolic executions, It would concentrate on the minutiae often overlooked when these approaches are used to smart contracts. More crucially, most of the discoveries and solutions for bias elimination are smart contract-specific, such as the suggested search depth range and the integration of multi-dimensional metrics. In contrast to previous work that advocated for benchmarks, It will illustrate that several additional elements might induce bias and provide a bias-free evaluation technique. Consider SmartBugs, which gives two datasets and focuses exclusively on the consistency of the execution outcomes without studying the execution process or suggesting a more effective assessment technique.

4.1.2. Setting up experiments and its overview

Security for smart contracts has been a hot topic of discussion in recent years, and relevant research papers have been published in a steady stream of high-quality publications. It has examined five highly referenced comprehensive surveys and tracked down citations to and from them to compile articles or tools linked to smart contract testing. There are 46 publications in all that come under this category; however, only 27 focus on vulnerability identification. Others focused on reverse engineering, interpretation and visualization, and gas optimization, among other topics not covered in this study.

I am interested in evaluating the experimental procedure for smart contract testing tools in this study. As previously stated, the conventional assessment procedure should consist of four phases to compare to past work. The details of each task in these phases are summarized in Table 3. It specifies the detection approach used for each tool, whether the tool is publicly available, which dataset served as the benchmark suite, which tools served as a baseline, and the depth limitation specified for every contract throughout operation. the number of tests carried for each setup, the delay time between trials, the sample size used to verify bugs, the indicators used to group related bugs, and the achieved level of code coverage.

Throughout the review process, it was determined that each piece of work on the list was superior to specified baselines or earlier work. However, due to the proliferation of empirical research on smart contract testing tools, the results in certain articles appear to be skewed, if not entirely contradictory to the actual scenario. This inconsistency is due to the absence of a standardized benchmark suite, scientific review procedure, and general performance measures. This paper shall explain the relevance of these elements in this study through comprehensive experiments and make recommendations for further research.

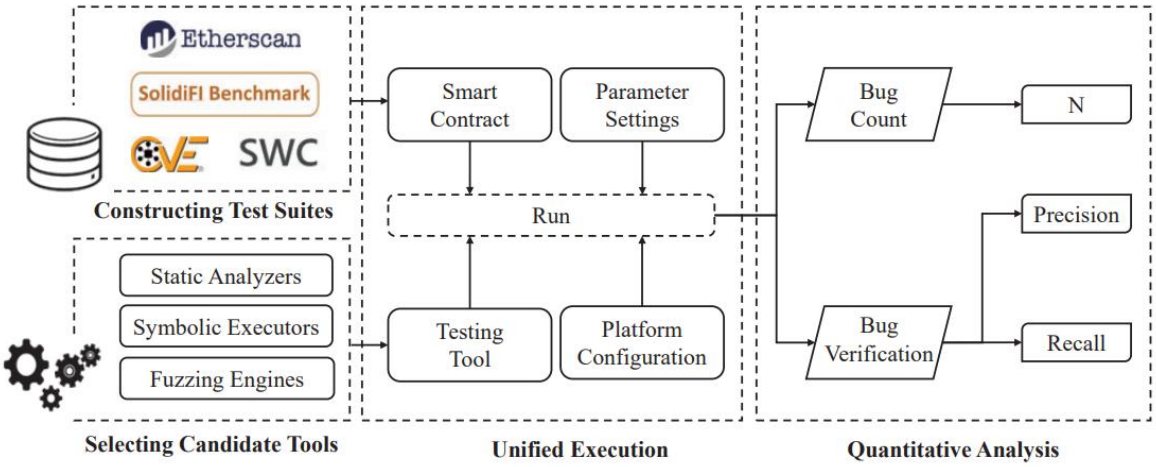


Figure 10: Overall experimental evaluation process (20)

The remainder of this section will describe the experimental setup and technique. Fig. 10 illustrates the whole procedure. To begin, I create unified test suites and evaluate potential testing technologies. Then, based on the tools, I filter and determine runtime settings. Following that, each tool begins analyzing the contract and generating problem reports in a unified execution environment. Finally, I collect the total number of alerts, carefully verify their validity, screen out any overlooked vulnerabilities, and determine the value of relevant metrics.

4.1.2.1. Constructing Test Suites

Due to the lack of common benchmarks for C/C++ applications, such as LAVA-M and Google fuzzer test suite, researchers are forced to scan real-world smart contracts from the Ethereum network or experiment with susceptible contracts put on GitHub. However, this approach mostly has two drawbacks: 1) Because Ethereum does not allow direct access to all contracts, repeated crawling is inefficient in terms of both personnel and time. Meanwhile, the extent to which these contracts may include vulnerabilities is unknown. Furthermore, without explicit labels, determining the legitimacy of detection findings is extremely difficult, and obtaining missing reports is impossible; 2) While the susceptible contracts downloaded from GitHub have clear labels given by experts, they are frequently brief and lack complex business logic.

Category	# Contracts	# Bugs	# Bug Types	# Reentrancy
UR	45622	-	-	10
MI	350	≥ 9369	7	119
CV	214	≥ 214	35	10

Table 2: Smart contract benchmarks and its analysis

Where UR-unlabeled real-world contracts;

MI-manually injected bugs;

CV-vulnerable contracts.

To compensate for the lack of a standardized test set, I created a benchmark suite using contracts crawled from Etherscan, SolidiFI, the Common Vulnerabilities and Exposures library, and the Smart Contract Weakness Classification and Test Cases library. They fall into three distinct categories: 1) real-world contracts that are not labeled; 2) contracts that have been intentionally injected with defects; and 3) confirmed susceptible contracts, as indicated in Table 2. Due to the magnitude and diversity of the benchmarks, I anticipate the evaluation results will generalize. To conduct additional research, I expose and confirm the reentrant code components buried in each contract in advance with the aid of two experts in the field of smart contract security. There are ten, one hundred and ninety-nine, and ten reentry vulnerabilities, respectively.

tool	method	benchmark	baseline	depth	trials	timeout	samples	bugs	coverage
ClairvOyance	SA	E(17700)	O, S, SL, M, SF	-	-		>1000	N, TP, FP, FN	
CONFUZZIUS	DF	E(27)	H, I, O, M, MT, OS	-	24	1H	100	N, TP, FP	I
ContractFuzzer	DF	E(6991)	O	-	1	80H	500	N, TP, FP, FN	
ContractWard	SA, DL	E(14850)	O, S	-	-		>1000	N, TP, FP, FN, TN	
DefectChecker	SE	E(587), B(165621)	O,M,S		10		500	N, TP, FP, FN, TN	I
EtherTrust	SA		O	-	-				
Harvey	DF	E(27)		-	24	1H	100	N	P, I
HoneyBadger	SE	E(151935)		50		30M	500	N, TP, FP	P
ILF	DF, DL	E(18496)	MA, CF	-		3H	>1000	N, TP, FP	I
MadMax	SA	E(91800)		-	-	20S	100	N, TP, FP	
MAIAN	SE	B(970898)		3		300S	>1000	N, TP, FP	
Manticore	SE	B(100)				90M			P
Mythril	SA, SE			22		24H			
Osiris	SE	E(1383), B(1207335)	Z	50		30M	500	N, TP, FP	P
Oyente	SE	B(19366)		50		30M	500	N, TP, FP	P
ReGuard	DF	E(5)	O	-	?	20M	10	N, FP, FN	
SASC	SA, SE	E(2952)	O			10M		N	P, BL, I

tool	method	benchmark	baseline	depth	trials	timeout	samples	bugs	coverage
sCompile	SE	E(36099)	O, MA	3		1M	>1000	N, TP, FP	P
Securify	SA	E(24594), U(100)	O, M	-	-		100	N, TP, FP, FN	
sFuzz	DF	E(4112)	O, CF	-	3	2M	500	N, TP, FP	BR
S-gram	SA, DL	E(1500)	RG	-	-			N, TP, FP	
Slither	SA	E(1000)	S, SC	-	-	2M	1000	N, TP, FP	
SmartCheck	SA	E(4603)	O,S,M	-	-		10	N, TP, FP, FN	
SolidityCheck	SA	E(1363)	M, O, S, SC, CF, OS	-	-		>1000	N, TP, FP, FN	
VeriSmart	SA	C(60), E(25)	O, M, OS, MT, Z	-	-	30M	100	N, TP, FP, FN, TN	
VerX	SE	E(138)	O, M, MT	5		1H	100	N, TP, FP, FN	
Zeus	SA	E(1524)	O	-	-	1M	>1000	N, TP, FP, FN	

Table 3: lists published assessments of smart contract testing tools (21)

Where SA- static analysis; DF – fuzzing; SE - symbolic execution; DL for deep learning

- *E - Etherscan; B – blockchain; U denotes user-written contracts; C denotes the database of Common Vulnerabilities and Exposures.*
- *The number in brackets denotes the total number of contracts evaluated. Baseline: O - Oyente, M - Mythril, S - Securify, H -Harvey, I - ILF, SL - Slither, SC - SmartCheck, SF - sFuzz, MA - MAIAN, MT - Manticore, CF - ContractFuzzer, RG - ReGuard, OS - Osiris, and Z - Zeus.*
- *TP - true positives, FP - false positives, FN - false negatives, and TN - true negatives. Coverage: I, P, BL, and BR denote instruction/path/basic-block/branch coverage, respectively.*

UR dataset collection and processing. The first category is based on data collected by Etherscan, the world's biggest decentralized platform for Ethereum smart contracts. Because all contracts on Etherscan are uniquely addressed, I begin by retrieving the addresses of all contracts with multiple transactions through Google BigQuery. For example, I retrieve 1,511,925 different contract addresses with the following query request:

```
SELECT contracts.address AS tx_count
FROM `bigquery-public-data.ethereum_block-chain.
     contracts` AS contracts
JOIN `bigquery-public-data.ethereum_block-chain.
     transactions` AS transactions ON (transactions.
     to_address = contracts.address)
GROUP BY contracts.address
HAVING tx_count > 1
ORDER BY tx_count DESC
```

Figure 11: Query to retrieve data from Google BigQuery

Then, using Etherscan's API, I construct a script to acquire the bytecode and/or source code. Finally, I gather 765,928 contracts that are entirely composed of bytecode, 4,063 contracts that are entirely composed of source code, and 741,934 contracts that are composed both of source code and bytecode. I preserve only source-available contracts for further analysis2. Then, I apply the approach described to eliminate duplicates, which involves comparing the MD5 checksums of the two source files after eliminating all blank lines and comments. I end up with 45,622 separate contracts after deduplication. Next, I added certain labels to the UR dataset to validate the legitimacy of defects reported by each tool. Specifically, I evaluate these contracts using all available techniques and extract 8,200 suspected susceptible ones. Finally, I undertake a two-month manual assessment with two specialists from the industry colleague WeBank and identify ten genuine vulnerabilities that may be exploited.

Real contracts typically have a greater number of lines of code and a more complicated underlying logic. Additionally, the availability of interface calls and inheritance complicates the interactions between contracts. These contracts' potential weaknesses are unknown, and they are frequently concealed in deeper branches, making them difficult to uncover. Additionally, the UR category contains a particular pattern of vulnerabilities, and tools can build tailored techniques to

improve performance. As a result, I require additional datasets with a greater variety of problem types to avoid overfitting.

Collecting and processing MI datasets. The second category includes SolidiFI, a manually generated benchmark created to aid academic research. It is a big library built by inserting bugs into several potential areas to create susceptible contracts with specific security flaws. SolidiFI supports seven specific bugs, including reentrancy, timestamp dependency, unhandled exceptions, unchecked send, transaction order dependency, integer overflow/underflow, and usage of tx.origin. Each category has 50 unique bug-free contracts. However, following injection, 9,369 unique flaws are distributed throughout all contracts.

These vulnerabilities are self-contained and do not affect other internal activities. Furthermore, their coding logic is straightforward, relying on only a few known vulnerability patterns. Along with the changed contracts dataset, the repository provides injection logs that may be used to pinpoint the position and nature of each problem.

Collecting and analyzing CV datasets. Finally, susceptible contracts that CVE reviewers or the SWC registry have validated are included in this category. To begin, I use the keyword "smart contract" to query the CVE database, which returns 513 results. Next, I download all source code files and categorize them into six categories, as defined, including arithmetic overflow/underflow, poor randomization, access control, and dangerous input reliance. Next, I acquired 124 different contracts after eliminating duplicates. Following that, I visit the SWC registry's official website and crawl 90 contracts associated with 37 unique IDs. Finally, I merge these vulnerable contracts to form a dataset size of 214.

Each contract has professionally validated vulnerabilities that are explicitly categorized and labeled. Additionally, the average number of code lines per contract in this dataset is less than 200, and there are no interface calls or inter-contract interactions.

4.1.2.2. Selecting Candidate Tools

Nine of the most commonly compared open-source programs were employed in the investigation, three for each detection approach (see Table 4). I use Security, SmartCheck, and Slither for static analysis; Oyente, Mythril, and Osiris for symbolic execution; and ContractFuzzer, sFuzz, and ILF for dynamic fuzzing.

Tools	Method	#Baseline	#Citation	Publication
Securify	SA	7	267	CCS'18
SmartCheck	SA	2	147	WETSEB'18
Slither	SA	1	43	WETSEB'19
Oyente	SE	16	1008	CCS'16
Mythril	SE	10	1500	White Paper
Osiris	SE	3	56	ACSAC'18
ContractFuzzer	DF	3	140	ASE'18
sFuzz	DF	1	8	ICSE'20
ILF	DF	1	22	CCS'19

Table 4: list of selected smart contract testing tools (21)

where

#Baseline denotes the number of times the tool has been used as a baseline

#Citation denotes the number of citations.

Since each tool is designed to address a specific issue, the sorts of vulnerabilities it covers differ significantly. To provide a fair comparison, I look at the sorts of vulnerabilities that each prospective tool supports. Then I map them to a list of the most often reported vulnerability categories for EVM-based smart contracts. Finally, I see that all of them support only reentrancy. Table 5 has a summary.

4.1.2.3. Unified Execution

A consistent execution environment is critical for dynamic executors, providing a single platform and consistent runtime parameters. I use the same value for each tool's corresponding parameter and maintain consistency with the other environment variables in each set of parameters experiment. Given that each tool might have an arbitrary number of self-defined parameters, I

conduct tests using the common parameters that they all support for tools that use the same implementation approach. Additionally, because the fuzzing technique is random, I provide the number of trials as a parameter. This paper's final parameter list includes depth limit, trials, and timeout.

Tool	Vulnerability types							
	AC	IO	DS	TO	RE	TM	UE	LE
Security	Yes			Yes	Yes		Yes	Yes
SmartCheck	Yes	Yes	Yes		Yes	Yes	Yes	Yes
Slither	Yes		Yes		Yes	Yes	Yes	Yes
Oyente	Yes	Yes	Yes	Yes	Yes	Yes		
Mythril	Yes	Yes		Yes	Yes	Yes	Yes	Yes
Osiris		Yes	Yes		Yes	Yes		
ContractFuzzer	Yes				Yes	Yes	Yes	Yes
sFuzz	Yes	yes			Yes	Yes	Yes	Yes
ILF	Yes				Yes	Yes	Yes	Yes

Table 5: A overview of the vulnerability categories that candidate tools offer (22)

For each form of vulnerability:

AC - access control;

IO - integer overflow;

DS - denial of service;

TO - transaction order dependency;

RE - reentrancy;

TM - time manipulation;

UE - an unhandled exception

LE - locked ether.

We prepare (at least) twenty different values for each of the parameters. The default value is 1, and the maximum value is 2X the mode value3 that is used by the majority of tools to configure this parameter. Due to the fact that the first two parameters are only used to evaluate symbolic executors or dynamic fuzzers, the interval between each pair of values is pre-defined. Timeout is a parameter that is shared by all tools. We selected increasing intervals in different ranges to account for the disparity in execution speeds across the various tools. For instance, the time interval between the first 5 seconds is 2 seconds; from the 5th to the 30th second, the interval is 5 seconds; from the 30th second to the 3rd minute, the interval is 30 seconds; from the 3rd minute to the 30th minute, the interval is 30 seconds; from the 30th minute to the 30th minute, the

interval is 5 minutes; and after the 30th minute, the interval is 30 minutes. Each runtime parameter is configured in detail in Table 6.

Parameters	Method	Mode	Maximum	Time Interval
Limit Depth	SE	50	100	5
Trials	DF	24	48	-
Timeout in seconds	SA,SE,DF	1800	3600	2,5,30,1800

Table 6: parameter in the experiment

All studies were conducted on three different machines. Each has eight cores (Intel i7-7700HQ @3.6GHz), 24GB of RAM, and runs Ubuntu 16.04.6 as the host operating system. In order to eliminate any differences between these computers, each proposed tool is always evaluated on the same computer.

4.1.2.4. Quantitative Analysis

I use unique bugs, accuracy, and recall as performance measures. Each tool generates an output file containing the analysis findings in JSON format when the execution is complete. Then, I parse it using a Python script, creating a mapping between vulnerability kind and line numbers. I can determine the total number of reported bugs (Ns), true positives (TPs), false positives (FPs), and missing samples (FNs) for each tool on each contract by comparing them to manually marked labels. I may calculate each measure using the following ways using these data:

Unique bugs. It relates to the number of problems reported by each tool after deduplication on each contract. I consider multiple reports of the same type of vulnerability on the same line to be duplicated and maintain only one count. After deduplication, the total number of bugs equals the number of unique bugs (N). I may get unique true positives (TP), false positives (FP), and missing samples (FN) using the same procedure, which is then employed in the following calculations.

Precision. It is defined as the ratio of genuine positives to all positive predictions made by a model. A higher accuracy number suggests a greater proportion of valid alarms and fewer false alarms, making bug verification and code change easier.

$$Precision = \frac{TP}{TP + FP}$$

Recall. It is the proportion of all relevant instances that were retrieved. A greater recall value means that more genuine vulnerabilities may be discovered, that fewer concealed vulnerabilities are overlooked, and that the danger of unknown attacks is reduced. Particularly, accuracy and recall are indifferent to size, implying that the paper's results will be unaffected by the absolute number of seeded defects.

$$Recall = \frac{TP}{TP + FN}$$

4.1.3. Analysis of results

This section will give experimental data and assess how tool performance varies among test suites, runtime settings, and evaluation measures. By observing the trends across many experimental settings, I uncovered some issues that had been overlooked in prior analyses. Then, I will recommend some principles for future research that should be followed when comparing to different baselines.

4.1.3.1. Variation on Different Test Suites

Recent research has used diverse datasets as test suites, including real-world contracts crawled from Etherscan and susceptible contracts validated by the CVE organization. However, none of the current benchmark options matched the comprehensiveness criterion. As seen in the fourth column of Table 3, 78 percent of tools (21 of 27) evaluate only one dataset type. Because the code features are confined in this case, the tool may readily design certain focused detection rules to improve performance by over-fitting. As a result, the findings generated from these trials may be biased and may not accurately reflect the tool's capabilities in all cases.

Fig. 12 illustrates how each measure varies across different test suites. I run each tool 48 times for each dataset to obtain final reliable findings. Most tools assessed (6/9) are deterministic, with each run producing the same output. The other tools converge for 12 hours. From the horizontal axis, it can be observed how each tool's performance differs significantly between

datasets. Following statistical analysis, It demonstrates that most differences are statistically significant ($p<0.021$ in the Wilcoxon test). Different dataset selections can result in wildly divergent experimental outcomes. Using SmartCheck and Slither as examples, Slither reports that SmartCheck generated a high number of false positives owing to a lack of in-depth knowledge of Solidity. The investigation demonstrates, however, that this conclusion is not always correct. SmartCheck accurately identifies ten existing vulnerabilities on the UR dataset and achieves an accuracy rate of 100%, which is 5X greater than Slither. When it comes to the MI dataset, they are almost identical. However, Slither outperforms SmartCheck by 1.6X on the final dataset.

Similarly, Mythril and Osiris exhibit the same scenario. Whether on real-world contracts or buggy contracts loaded with manufactured bugs, Osiris can detect more hidden flaws with greater accuracy, as seen by a 10% improvement in precision and a 30% increase in recall. However, the observation is fully inverted on the CV dataset, where the code structures are much simpler. As a result, while Osiris discovers six additional defects in total, it lags Mythril by ten percentage points in both precision and recall.

Deeper Analysis. The inability of tools to execute reliably is mostly due to targeted rule designation for a certain vulnerability pattern and insufficient processing capability for deep calls. Additionally, differing judgment standards for genuine vulnerabilities result in disparate evaluations of tool performance. Only vulnerabilities that can be exploited in practice are used as positive samples in this article, which is more practical than the judgment requirements used in prior publications but results in a loss of precision. Take Slither as an example; it will flag addr.transfer() as a reentrancy error since it includes a built-in gas limit of 2300 to prevent reentry difficulties. Since these false-positive samples were classified as actual bugs during the first evaluation of the published tools, the majority of them (22/27) reported a high accuracy rate in their papers but performed badly in practice.

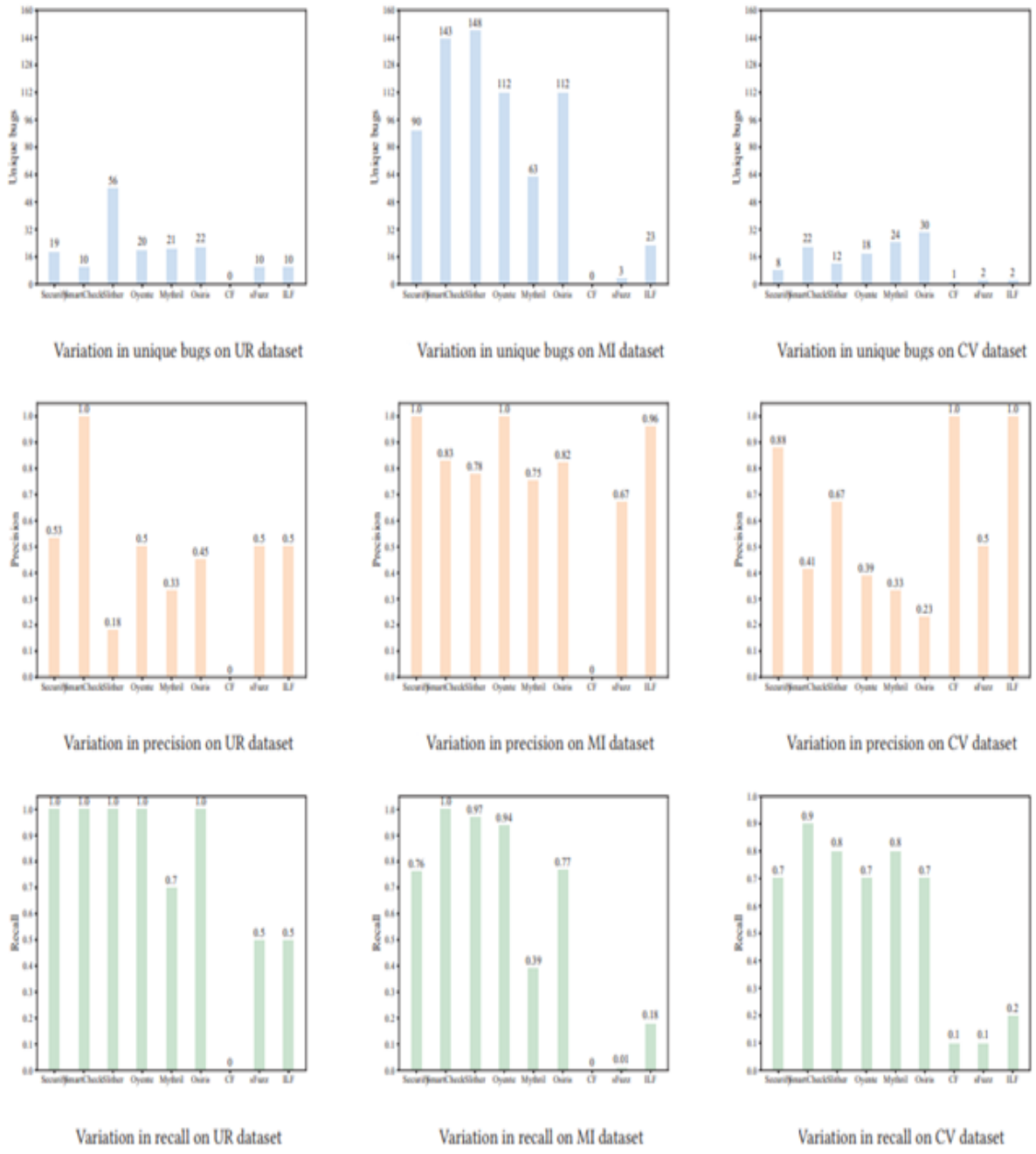
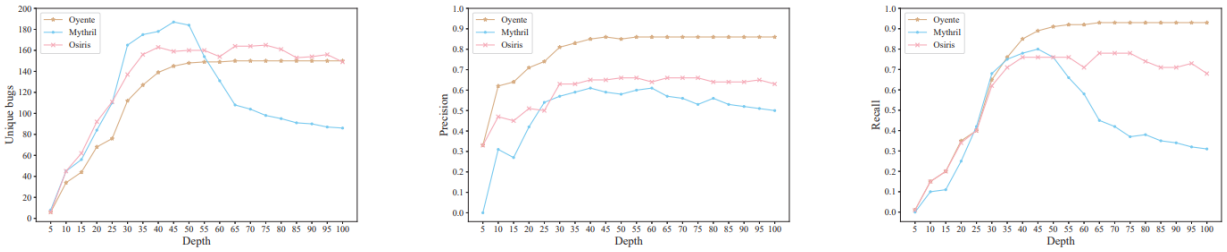


Figure 12: Variation in unique bugs, precision, and recall on different test suites of all tools (Excel)

4.1.3.2. Variation on Runtime Parameters



Unique bugs vs. depth limit | Precision vs. depth limit | Recall vs. depth limit

Figure 13: Variation in unique bugs, precision and recall with different depth limit of Oyente, Mythril and Osiris (Excel)

This section will empirically illustrate various factors' effects on the performance of tools4. A tool's execution is dependent on its execution environment, which consists of a generic platform and the appropriate runtime parameters. For static tools, they frequently start with a bytecode-based intermediate expression and then do pattern matching or data structure analysis on it. Thus, after installing the necessary dependencies, the user simply has to give the contract's source code or bytecode and the maximum execution duration. Dynamic tools are more sophisticated since they need the user to specify extra runtime parameters, such as the maximum recursion depth. Most users utilize the same parameter settings throughout the assessment process for all tools. However, this may not be the best option. Experiments conducted with parameters that are inappropriate for them may yield conflicting results. The details are included below.

Adjust the Maximum Search Depth: The maximum depth that may be attained via a single path is referred to as the depth limit. Most symbolic executors access nodes in the control flow graph in a depth-first manner. Each instruction included within a basic block will be performed symbolically, beginning with the entry node. After completing the previous command, the executor will proceed to the next block based on the jump condition. This operation will continue until the termination block is reached or the search depth reaches its maximum value.

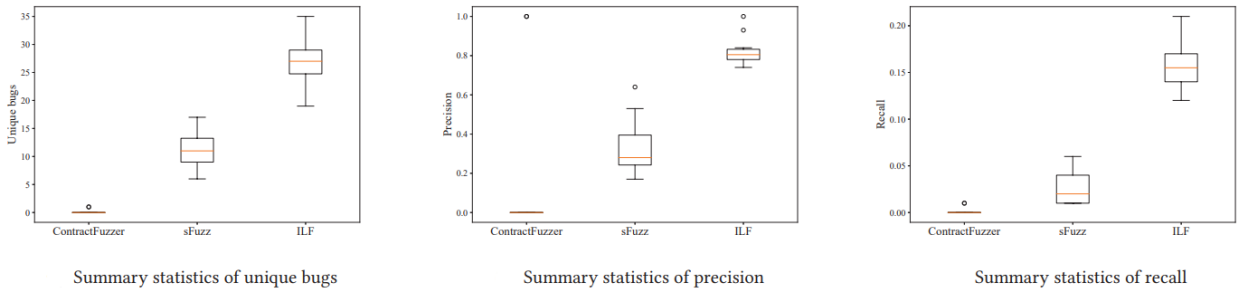
The depth constraint is crucial for finding deeper vulnerabilities. When it is set to a too big value, blocks on other execution routes become unavailable for a certain amount of time, resulting in the neglect of vulnerabilities. Varying symbolic execution tools have very different depth

settings. Among ten recent articles (see Table 3), MAIAN and sCompile specify a depth limit of three; VerX and Mythril specify five and twenty-two, respectively; Oyente and its derivatives, Osiris and HoneyBadger, specify a huge threshold of fifty; and others do not specify a depth restriction at all.

I evaluated Oyente, Mythril, and Osiris at various depth limitations, and the resulting performance differences are depicted in Fig. 13. As search depth rises, the path that each tool may access gets more complex, and the number of reported problems, precision, and recall continues to climb. However, when the search depth surpasses 50, it begins to impair the tool's usual operation. Due to the excessive time spent on deep blocks of a single path, the performance of each tool degrades in variable degrees. Using a variety of various depth limitations for examination might result in a variety of different results. When the maximum search depth is set to 25, Mythril and Osiris both report 110 separate defects, but Mythril outperforms Osiris by having a 4% better accuracy rate and four extra flaws. When the depth limit is increased to 70, Mythril's performance lags significantly behind Osiris's. This time around, Osiris reports 164 specific bugs with an accuracy rate of 66% and a recall rate of 78%. At the same time, Mythril can only detect 104 alleged bugs with an accuracy of 56% and recall of 42%.

Increase Test Trials*:* As is well known, randomization significantly influences dynamic fuzzers. Therefore, it is insufficient to run each fuzzer once and compare their results during the assessment. A possible option is to conduct many rounds of experiments to conduct statistically valid comparisons.

This requirement should also apply to other fuzzing-based testing methods. Regrettably, Table 1 demonstrates that just three fuzzing tools execute tests several times.



Summary statistics of unique bugs | Summary statistics of precision | Summary statistics of recall

Figure 14: Variation in unique bugs, precision and recall throughout 48 trials of ContractFuzzer, sFuzz and ILF (Excel)

I ran ContractFuzzer, sFuzz, and ILF independently to ensure that the performance of each run was steady. Fig. 14 depicts the overall performance distribution of each tool throughout 48 distinct runs. The data distribution is split into four equal sections for each tool, and a box shows the highest and lower quartiles. The orange line within it denotes the median value, while the lines outside it indicate the minimum and highest values. Individual outliers, often highly good or negative points, are denoted by little dots. It has been discovered that even within the same execution environment, performance might vary significantly between runs. Consider ILF. It can detect up to 35 distinct bugs, approximately twice as many as the lowest point. Its highest precision rate is 100%, and it can identify up to one-fifth of genuine vulnerabilities.

Meanwhile, its accuracy rate dips to around 70% in certain rounds, and its recall rate is less than 10%. A single run's result is contingent, and the dominance relationship it reflects may be skewed. In practice, it is recommended to conduct numerous trials and analyze using the mean or the most often occurring values.

***Extend the Execution Timeout*:** The execution timeout specifies the maximum amount of time a tool may spend analyzing a contract. It is a user-customizable parameter for each tool. A widely held belief is that a longer time period may reveal a more steady performance trend. Meanwhile, it is widely considered that increasing execution time beyond a certain point is unnecessary and leads to resource waste. The rationale for this is that they will only produce minor swings in performance and will have no effect on the tool's dominance relationship. These two assumptions offer considerable leeway for configuring the execution timeout. The column timeout in Table 3 demonstrates the enormous variation in timeout settings among previous assessments,

ranging from 20 seconds to more than three days. The most often used time period is 30 minutes, employed in four articles. Nearly 70% of the remaining users specified an arbitrary timeout value between 2 minutes and 1 hour. MadMax, sCompile, and Zeus all have a timeout of fewer than two minutes, even though the majority of them are static tools. The following four publications run their instruments for more than an hour, with each tool being dynamic.

I set up 20 termination times and conducted trials on all selected tools to determine the effect of execution timeout on tool performance. Fig. 15 illustrates the trends. Due to the execution speed disparity, dynamic tools sometimes take longer to notify errors and achieve stability. ContractFuzzer, for example, did not access the branch containing the problem until the 1800th second. However, due to the fact that this is the only known problem, the accuracy leaps from 0% to 100% and stays there.

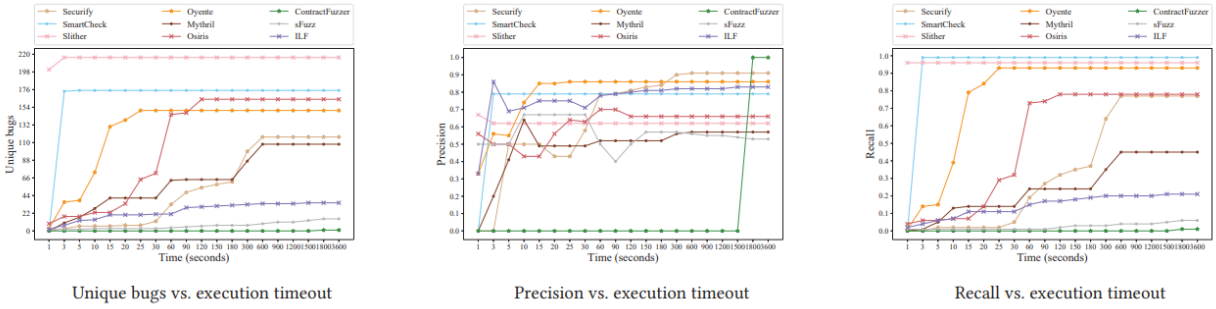


Figure 15: Variation in unique bugs, precision and recall with different execution timeout of all tools (Excel)

In comparison, SmartCheck completed the examination of the whole code in less than two seconds, and the precision remained constant after that. Prior to reaching the upper bound, performance improvement is non-linear. As a result, the relative performance of tools will fluctuate over time. In this case, a brief wait for review may result in an imprecise judgment. For example, when Mythril and Osiris are run with a timeout of 10 seconds, Mythril reports a total of 28, five more than Osiris. Mythril also outperforms Osiris in accuracy and recall by 21% and 6%, respectively. However, running them for a longer period of time appears to tell a different narrative. When the delay is set to 120 seconds, Osiris reports a total of 164 distinct defects, 109 of which are genuine, accounting for 78% of all concealed vulnerabilities. By comparison, Mythril

reports just 64 distinct flaws, of which half are genuine, accounting for fewer than a fifth of vulnerabilities.

Deeper Analysis. The greatest advantage that any tool's detection method can provide depends on how many execution settings are configured. As a result, even tools that perform the same approach may exhibit varying degrees of parameter sensitivity. For illustration, the optimal search depth limit for breadth-first tools may be larger than the optimal search depth limit for depth-first tools because the greater depth limit does not prevent breadth-first tools from discovering shallow vulnerabilities in other branches but prevents depth-first tools from accessing other branches.

4.1.3.3. Variation on Performance Metrics

In addition to benchmark suite and runtime parameters, the selection of performance metrics also needs to be persuasive and reasonable. Looking back on published work in recent years, there are two commonly used performance metrics: discovering real vulnerabilities and code coverage. The former can be subdivided into three metrics, the total number of bugs, precision, and recall; the latter generally has three manifestations. As shown in Table 3, the most commonly used is path coverage, which was used by six papers, referring to the coverage of all feasible paths in the code. Then comes instruction coverage used by five papers, which counts the number of executed instructions. The remaining two types are basic-block and branch coverage, calculated as the percentage of the covered basic blocks or branches. It seems to make sense that coverage can represent the ability of a tool to discover vulnerabilities: the more code a tool executes, the more likely it is to find vulnerabilities. However, the relationship between them does not seem to be strong. As a result, utilizing it directly to demonstrate the tool's bug-finding capability (such as Manticore) is similarly inadvisable and insufficient.

Consider the following example of a real-world contract named ItemToken5. I separately ran ILF and sFuzz and recorded the variation in instruction coverage6 and the total number of reported vulnerabilities over time. The results are shown in Fig. 16. After 30 seconds, neither ILF nor sFuzz discovered any new vulnerabilities, and coverage remained unchanged. Therefore, I drew the curves of the first 30 seconds to explain.

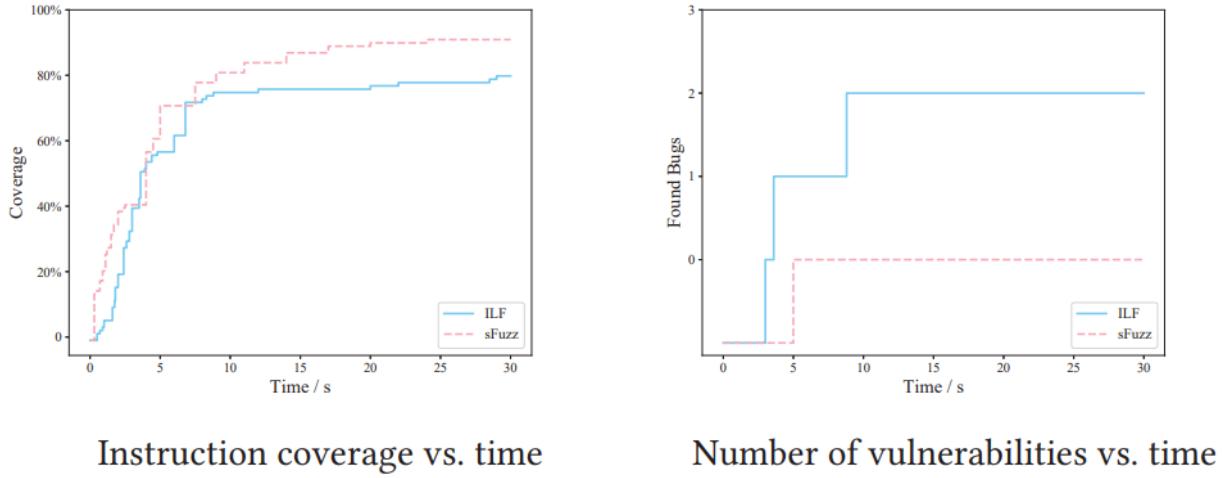


Figure 16: Variation in coverage and number of vulnerabilities reported by sFuzz and ILF over time (Excel)

Although sometimes, the increase in coverage is accompanied by the growth in vulnerabilities. For example, at the fifth second, sFuzz succeeded in entering a new branch, and the coverage increased from 61% to 71%, while a new vulnerability was discovered. Nevertheless, comparing the number of steps in (left) and (right), it can be found that most of the time, while the coverage increases, the number of vulnerabilities remains unchanged. For example, in sFuzz's coverage curve, only one out of 22 rises results in discovering new vulnerabilities. Similarly, ILF's coverage curve has increased 29 times, while the vulnerability finding curve has altered just three times.

What's more, higher coverage does not guarantee a stronger ability to find vulnerabilities. For example, as shown in Fig. 16 (left), sFuzz achieves a coverage rate of 91 percent, while ILF only achieves an 80 percent coverage rate. However, as shown from (right), ILF finally finds three unique bugs while sFuzz can only find one. So, in this case, it is not very objective to conclude that ILF performs better than sFuzz according to higher coverage.

Deeper Analysis. The most fundamental metric to measure tool performance is the number and percentage of real vulnerabilities it finds. As an indirect indicator, higher coverage cannot stand for a better performance of the tool. The increase in coverage may be caused by meaningless code, such as test code, which is not directly helpful to vulnerability detection.

Finding: Different tools have different performances on different metrics. To reach a rigorous conclusion, I need to evaluate more comprehensive metrics, including the number of unique bugs, precision, recall, and coverage.

4.2. Data Integrity Test

4.2.1. Experimental environment

Cloud data integrity protection has emerged as a significant research area at this point. While utilizing network coding technology for storage and integrity protection has achieved some results, the computational overhead associated with the integrity protection process is too high, adversely affecting the storage node's operational efficiency and reducing the cloud storage service's availability. Therefore, this study conducts simulated tests to validate the accuracy of the initial theoretical proof. As seen in Table 7, the experimental setting is as follows.

Category	Parameter
CPU	Core i7(TM)2
Processor Frequency	2.66 GHz
RAM	8 GB
OS	Windows 10
Software Simulation	MATLAB 9.0

Table 7: Parameters of the environment during experiment

Procedure for carrying out experimental simulations (see Fig. 17): During the experimental simulation, a virtual cloud storage service environment is created, consisting of one cloud storage management server and five storage node servers, one of which serves as a backup node for the data integrity protection node and the remaining four as the operational cloud. Node for storage service. The cloud service management node holds the essential information about each data storage node and is primarily responsible for managing the related storage node; the storage node saves the relevant data and the link between the storage servers, mostly for data storage. Additionally, data transit is required for data integrity protection. The experimental simulation's unique environment is as follows. First, the k data blocks are distributed among n nodes to facilitate discussion of data integrity protection in a single node failure. To compare the computational

overhead of the two data integrity protection techniques, the computed quantity of data is the number of multiplication operations performed throughout the data integrity protection procedure.

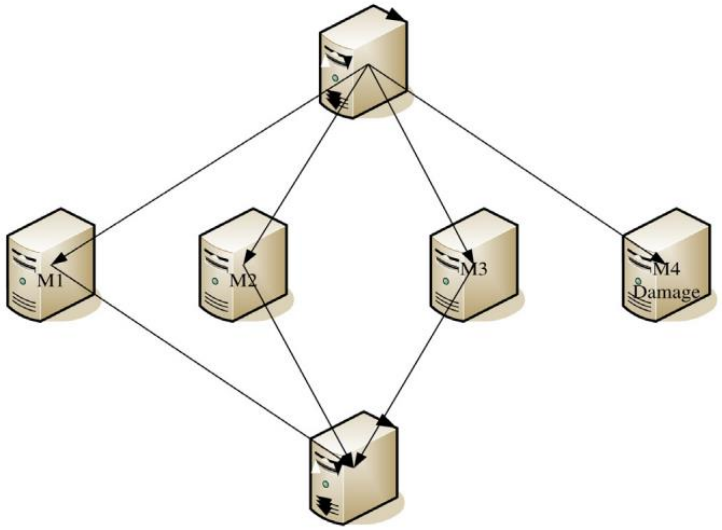


Figure 17: Topology of experimental simulation

4.2.2. Calculation in the program

This paper establishes a blockchain network by installing distributed virtual machine agents in the cloud and utilizing the cloud's multi-tenancy. A cloud data integrity verification technique called BPDP is studied. It consists of five algorithms/steps:

(1) Generating a public key/private key pair

KeyGen (1k) → (pk.SK)

(2) Generate a digital label

TagBlock (pk, SK, m) → Tm

(3) Formulate a piece of information to challenge the label

GenChal (c,r) → chal

(4) Generate evidence based on the previous steps

GenProof (pk, F, chal, Σ) → V

(5) Test evidence as shown in the formula

Check Proof (SK, V) → results

4.2.3. Integrity verification process

Fig. 18 depicts the integrity verification step. First, the user selects a data block at random, challenges the CSP's storage node through the VMA node, receives the file location specified in the IPFS cluster query block, creates proof, and returns it to the VMA. Next, the VMA performs verification in order to compute the evidence. If it is genuine and valid, the second step is confirmed. Finally, the MHT is used to determine the existence of the challenge block and its consistency with the root hash value. If they concur, the evidence file is complete; otherwise, it is deleted.

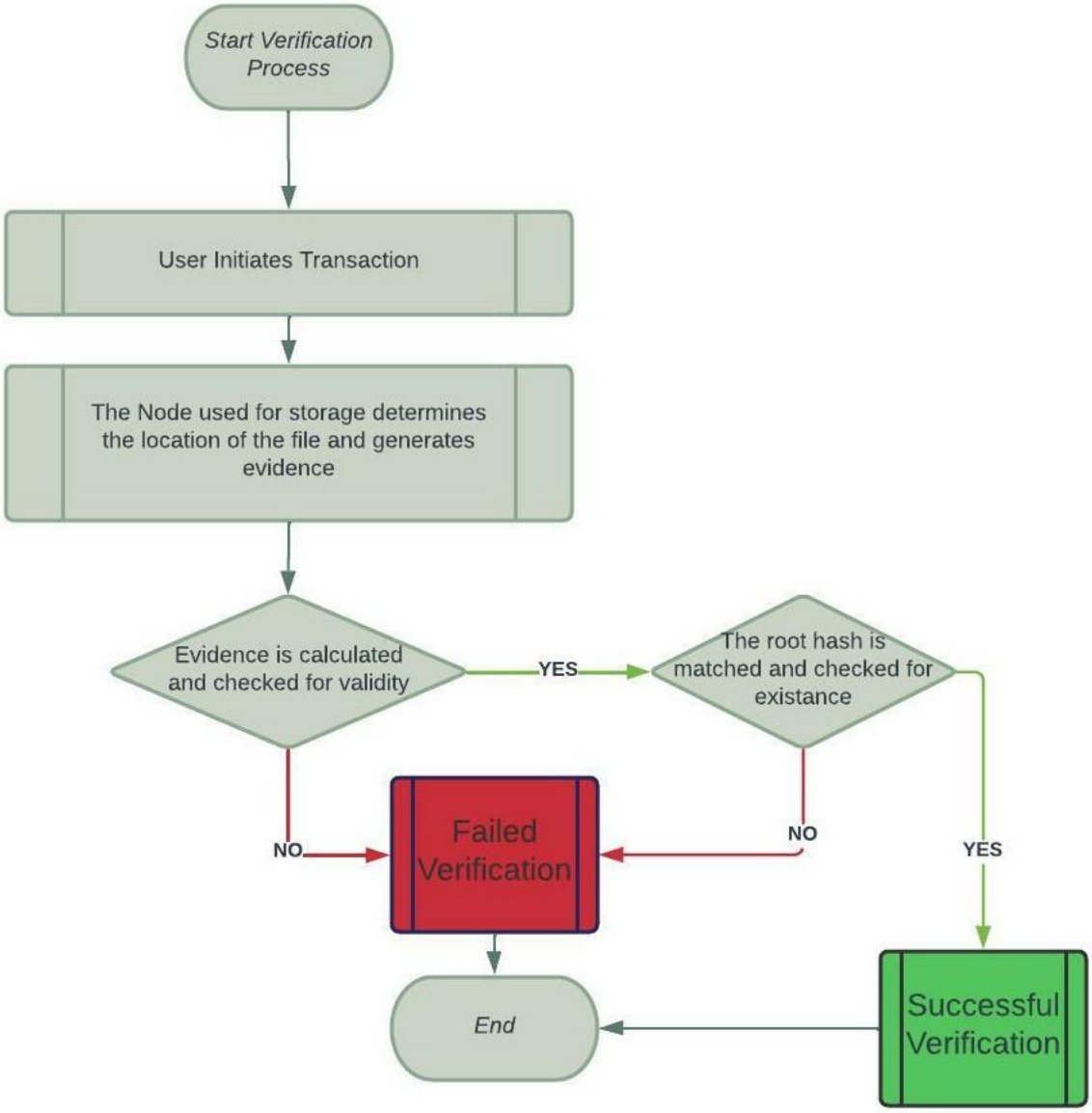


Figure 18: Verification of data integrity in a flowchart

4.2.4. Cloud data integrity analysis based on blockchain

To begin, this experiment assesses the protocol's correctness using sampling-based integrity verification. The total number of data blocks on the cloud server is considered to be n. If data is tampered with improperly, the ratio of corrupted data blocks is: pb = e n, assuming t is per second. Because the number of data blocks in the secondary challenge is proportional to the total number n, the likelihood of detecting illicit tampering is P, as defined in the Equation below.

$$P = P\{X \geq 1\} = 1 - \frac{n-e}{n} \cdot \frac{n-1-e}{n-1} \cdots \frac{n-t\cdot n-e}{n-t\cdot n} \geq 1 - \left(\frac{n-e}{n}\right)^{t\cdot n} = 1 - (1-P_b)^{t\cdot n}$$

4.2.5. Implementation of some functions

The purpose of this work is to construct a fixed-size file by randomly populating letters or integers. To pre-process the file, the user specifies the correct number of data segments, data segment size, and data block size. First, the file is separated into data blocks, which are then divided further. Finally, divided into data chunks, as seen in the illustration below.

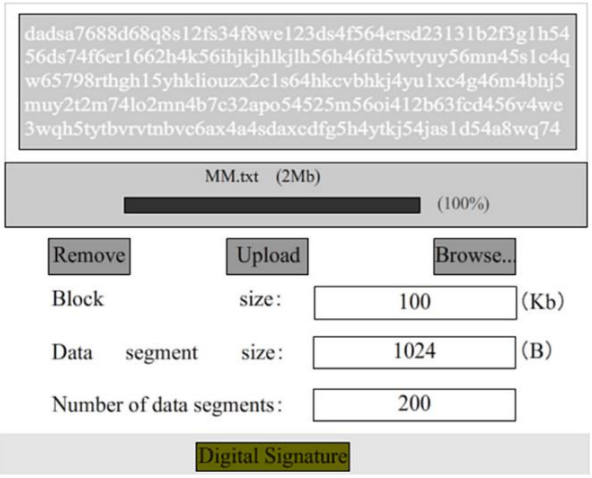


Figure 19: Sample pre-processing function

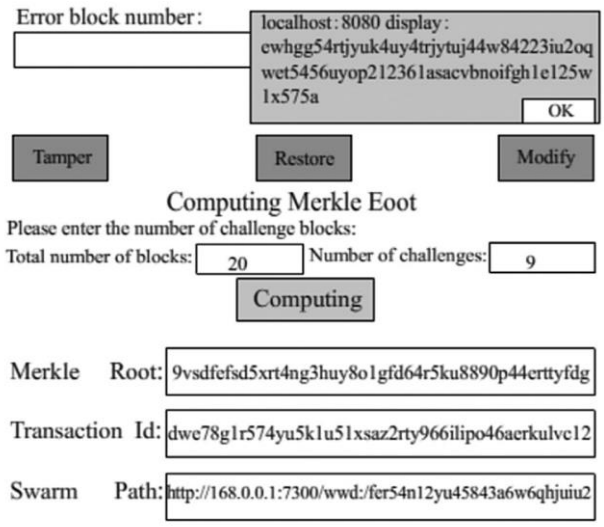


Figure 20: Data label destruction after verification

The Figure above illustrates the verification results of simulation data tag destruction. If the data stored in the storage node is destroyed, the data's digital label is likewise altered, and only the MHT root hash value is recalculated, and the information in the blockchain network is compared, and the file is destroyed and the new calculation completed. The MHT hash differs from the original MHT hash, suggesting that the data tag has been corrupted.

5. Discussion and Results

Though current testing tools can detect a significant amount of potential security vulnerabilities in actual projects, I still find out the following points worthy of being discussed:

A single assessment standard is important. I have investigated numerous related papers and analyzed their experimental methodologies in this paper. The description in the paper showed that they did not share a unified setting. Using nine representative tools, I illustrated that those individual evaluations with different experiment settings could lead to misleading conclusions. It can be found that:

- Over 70% of articles utilized just unlabeled real-world contracts as a benchmark, where the vulnerability distribution is unknown and the vulnerability pattern is straightforward. This makes it hard to conduct a full evaluation of the tool's capacity to detect all types of vulnerabilities and makes it simple to overfit, as demonstrated by SmartCheck's performance on the UR dataset. Simultaneously, without clearly labeled samples, it is also hard to reliably count false negatives. Experiments demonstrated that varying test suites have a significant effect on tool performance.
- Papers varied widely on the setting of runtime parameters, such as maximum search depth. However, the experiments showed that too large a threshold would hinder the normal execution of the tool.
- Most papers failed to take varying performance caused by runs into consideration. The experiments showed that it is necessary to perform multiple trials with statistical tests to balance the impact of randomness.
- Among all papers, the choice of execution timeout ranges from 20 seconds to 80 hours. However, the trials demonstrated that a longer wait is required to accurately depict a tool's performance, particularly for dynamic tools.
- Only little more than half of the publications assessed the tool's performance across all metrics: unique bugs, accuracy, and recall. Others chose just one or two to demonstrate their greatest tools, or even judged their effectiveness solely on the basis of coverage. Experiments demonstrated that using only a portion of them is insufficient.

- Runtime setting choices have a significant impact on the tool's performance. Therefore, to produce a more impartial and compelling assessment, it is recommended that acceptable and adequate values for each tool's custom parameters are supplied during the experiment.
- The test suite's code size and vulnerability pattern have a significant influence on tool performance. Therefore, it is preferable to examine a multi-type integrated benchmark suite to avoid biases.

6. Conclusion

Worldwide, commercial and non-profit use of blockchain technology have proved considerable advantages over existing arrangements. The technique appears to be particularly suitable for applications that demand the storing and processing of huge volumes of secure data. Effective technology research and experimentation in a range of sectors require a favorable legal framework. Additionally, it should be noted that the primary characteristic of the technology is that it represents a dispersed network with encrypted data and no central server that might be hacked or changed with. Among other things, blockchain-based arrangements such as automated auctions, smart contracts, and decentralized autonomous organizations represent significant advances toward more decentralisation.

It is impossible to see widespread use of blockchain technology in a centrally controlled economy. Instead, it is likely to be most successful under a system that favors little government engagement in the economy, with the regulator only setting and enforcing rules and serving as an adjudicator in disputes.

In practice, administering an administrative procedure entails entering information about one's civil status, property rights, and health into an official registry. As a result, blockchain technology may be considered a unique and universal technology that enables the automation and simplification of practically all administrative processes while boosting the transparency and efficacy of e-government.

7. References

1. **Nakamoto, S.** *Bitcoin: A Peer-to-Peer Electronic Cash System.* 2008.
2. **Pluralsight.** Blockchain Architecture. *Pluralsight.* [Online] 2017. https://www.pluralsight.com/guides/blockchain-architecture.
3. **CoinMarketCap.** Cryptocurrency Market Capitalizations. *CoinMarketCap.* [Online] https://coinmarketcap.com/..
4. **Dance, Coin.** [Online] Coin Dance.
5. *Blockchain is meaningless.* **Jeffries., A.** s.l. : https://www.theverge.com/2018/3/7/17091766/, 2018.
6. *The Path of the Blockchain Lexicon (and the Law).* **Walch., A.** 2017. SSRN 2940335.
7. **Blockchain and distributed ledger technologies. [Online] https://www.iso.org/committee/6266604/x/catalogue/. ISO/TC 307.**
8. **Sch□uritz, S. Seebacher and R. Blockchain Technology as an Enabler of Service Systems: A Structured Literature Review. [book auth.] M. Drgoicea, and M. Cavallari In S. Za.** ***Exploring Services Science,.*** **cham : Springer International Publishing,, 2017.**
9. **S. P. Stawicki, M. S. Firstenberg, and T. J. Papadimos.** ***What's new in academic medicine? Blockchain technology in health-care: Bigger, better, fairer, faster, and leaner.*** **2018.**
10. ***Bitcoin and Cryptocurrency Technologies. .*** **A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. s.l. : Princeton University Press, , 2016.**
11. ***Understanding the Blockchain Using Enterprise Ontology. In Advanced Information Systems Engineering, Lecture Notes in Computer Science.*** **Weigand, J. d. Kruijff and H. Cham : Springer, 2017.**
12. **Kahn, D.** ***The CodeBreakers.*** **s.l. : The Macmillan Company, 1973.**
13. **IBM Knowledge Center. Symmetric cryptography.** ***IBM.*** **[Online] https://www.ibm.com/support/knowledgecenter/en/SSB23S_1.1.0.14/gtps7/s7symm.html..**
14. **IBM Knowledge Center - . Public key cryptography.** ***IBM.*** **[Online] https://www.ibm.com/support/knowledgecenter/en/SSB23S_1.1.0.14/gtps7/s7pkey.html.**
15. **Bitcoin Energy Consumption Index.** ***Digiconomist.*** **[Online] https://digiconomist.net/bitcoin-energy-consumption.**
16. **L. Bell, W. J. Buchanan, J. Cameron, and O. Lo.** ***Applications of Blockchain Within Healthcare. Blockchain in Healthcare Today,.*** **2018.**
17. **Buterin., V.** ***A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM.***
18. **Wood., A. Antonopoulos and G.** ***Mastering Ethereum: Building Smart Contracts and Dapps.*** **s.l. : O'Reilly UK Ltd,, 2018.**
19. **Tracking blockchain technology and regulation around the world.** ***BitLegal.*** **[Online] http://bitlegal.io/..**
20. **Etherscan. [Online] Etherscan.io.**
21. **[Online] https://blog.logrocket.com/complete-guide-blockchain-testing/.**
22. **Marijan, Chhagan Lal and Dusica. [Online] https://arxiv.org/pdf/2103.10074.pdf.**
23. **G20. Communiqu G20 Finance Ministers & Central Bank Governors Meeting. [Online] 2018. https://g20.org/sites/default/files/media/communique-_fmcbg_july.pdf.**
24. **EU. European countries join Blockchain Partnership.** ***Europa.*** **[Online] https://ec.europa.eu/digital-single-market/en/news/european-countries-join-blockchain-partnership,.**
25. ***Handbook of applied cryptography.*** **J. Katz, A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. s.l. : CRC Press, 1996.**
26. **Nandwani., K. What is a Blockchain and why does it matter?** ***http://kunalnandwani.com/.*** **[Online] 2016. http://kunalnandwani.com/blockchain-matters/.**
27. **S. Voshmgir, Kalinov, and V. Kalinov. Blockchain Technology Handbook. [Online] https://s3.eu-west-2.amazonaws.com/blockchainhub.media/Blockchain+Technology+Handbook.pdf.**
28. **Software System Security Assurance Group — WingTecher Lab. [Online] http://wingtecher.com/bugs/cveen.**

8. List of Tables and Figures

8.1. List of Tables

8.2. List of Figures

8.3. List of Abbreviations

DAO - Decentralized Autonomous Organizations

MHT – Merkle Hash Tree

ICO - initial coin offering

EVM - Ethereum Virtual Machine

CVE - Common Vulnerabilities and Exposures