A picture containing text, font, graphics, screenshot

Description automatically generated

**Web Security – IE2062**

**Assignment**

**Smart Contractor**

**Table of Contents**

[Acknowledgement 3](#_Toc136196116)

[What is a smart contractor? 4](#_Toc136196117)

[Benefits of smart contracts 5](#_Toc136196118)

[Disadvantages of smart contracts 5](#_Toc136196119)

[History of smart contractor 5](#_Toc136196120)

[Smart contracts uses 6](#_Toc136196121)

[Pros and Cons of smart contracts 7](#_Toc136196122)

[Common issues and challenges with smart contracts 7](#_Toc136196123)

[Smart contract development platforms 8](#_Toc136196124)

[Most common smart contract vulnerabilities 9](#_Toc136196125)

[Various steps for building a smart contract 11](#_Toc136196126)

[Are smart contracts Enforceable? 13](#_Toc136196127)

[Smart contracts can be created without coding 13](#_Toc136196128)

[Applications of smart contracts 14](#_Toc136196129)

[Smart contracts use cases 15](#_Toc136196130)

[Future of smart contracts 16](#_Toc136196131)

[References 17](#_Toc136196132)

# Acknowledgement

Before I start writing my assignment, I'd like to sincerely thank Dr. Lakmal Rupasinghe, the lecture incharge, for his guidance and help in getting our entire class started on and complete our assignments.

I also want to convey my sincere gratitude to Ms. Chethana Liyanapathirana for the guidance and constant support she gave me and the rest of our class as we worked to complete this task.

# What is a smart contractor?

A decentralized program known as a smart contract runs business logic in response to events. Executing a smart contract can lead to an exchange of funds, the provision of services, the release of digitally locked content, or other sorts of data manipulation, such as changing the name on a land title. Additionally, smart contracts can be used to enforce privacy protection by, for instance, enabling the selective release of data that is privacy-protected in order to comply with a particular request.

The programs that support smart contracts can be created, distributed, managed, and updated using a number of architectures. They can be included in different payment methods and digital exchanges, including those that accept bitcoin and other cryptocurrencies, and stored as a component of a blockchain or other distributed ledger technology.

Smart contracts, despite their name, are not enforceable contracts. Their primary duty is to programmatically carry out business logic, which has been programmed into them to carry out certain tasks, processes, or transactions in response to a specific set of criteria. To connect this execution to legally enforceable agreements between parties, legal action must be done [1].

## **Benefits of smart contracts**

* Speed, efficiency and accuracy

The contract is promptly carried out if a condition is satisfied. Smart contracts are digital and automated, so there is no paperwork to complete, or time spent fixing mistakes that frequently occur when documents are filled out manually.

* Trust and transparency

There is no need to wonder whether information has been changed for one participant's personal gain because there is no third party engaged and participants share encrypted records of transactions.

* Security

Because the blockchain transaction records are encrypted, they are incredibly difficult to hack. Additionally, hackers would need to alter the entire chain in order to change a single record on a distributed ledger because each record is linked to the records that came before and after it.

* Savings

Smart contracts do away with the need for middlemen to conduct transactions, along with the costs and wait times that go along with them [2].

# Disadvantages of smart contracts

* Difficulties to change

Due to their immutability, smart contracts are subject to conflict with already-existing transactions and must be changed through new transactions. Due to the fact that the code was originally written by blockchain developers, there are also technical mistakes and flaws.

* Scale problems

Due to smart contracts replicating every transaction to all nodes, blockchain transactions can take some time. In a fully functional network, the Ethereum blockchain can only process fifteen trades in a second and thousands of transactions a minute. Congestion causes the transaction to take longer than intended and increases costs [3].

# History of smart contractor

Nick Szabo, an American computer scientist who created the virtual currency "Bit Gold" in 1998, ten years prior to the launch of Bitcoin, first advocated smart contracts in 1994. In truth, Szabo has refuted claims that he is the genuine Satoshi Nakamoto, the person who created Bitcoin anonymously.

Smart contracts, as described by Szabo, are computerized transaction protocols that carry out a contract's terms. He desired to bring POS (point of sale) and other electronic transaction techniques' capability into the digital sphere.

Szabo also advocated for the execution of a contract for synthetic assets, including derivatives and bonds, in his paper. According to Szabo, "These new assets are generated by combining securities (such as bonds) and derivatives (such as options and futures) in numerous ways. Due to computerized analysis of these complicated term structures, very complex payment term structures can now be included into conventional contracts and exchanged with low transaction costs.

In ways that were not possible with blockchain technology, several of Szabo's predictions from the paper came true. For instance, the majority of today's trading in derivatives is done via computer networks using sophisticated term structures [4].

# Smart contracts uses

Smart contracts can be used for a variety of things since they implement agreements. Assuring that transactions between two parties, including the purchase and delivery of commodities, take place is one of the simplest uses. For instance, a company in need of raw materials may use smart contracts to arrange payments, and the supplier could arrange supplies. The cash may then be automatically sent to the supplier upon shipment or delivery, depending on the terms of the agreement between the two organizations.

Smart contracts can be utilized in a variety of contexts, including real estate transactions, stock and commodity trading, loans, corporate governance, supply chains, dispute resolution, and healthcare [5].

# Pros and Cons of smart contracts

Pros:

Smart contracts have the advantages of being traceable, transparent, and irrevocable. This implies that a smart contract cannot be altered after it has been created. All parties involved may see the contract and how it will be carried out, and its conditions are unchangeable. Additionally, the smart contract is irreversible, which means that once it is started, there is no turning back.

Cons:

Cons of smart contracts include their complexity and potential for errors and/or flaws. This is so because people write smart contracts, and we all know that people make mistakes! Finding the problem and fixing it if a smart contract malfunctions might be challenging. Additionally, smart contracts are not always available to everyone because they are digital and rely on code. Internet connectivity and computer/smartphone proficiency are not universally available.

This should have provided some insight into the subject of smart contracts. We talked about their history, purpose, mechanics, and some advantages and disadvantages [6].

# Common issues and challenges with smart contracts

* Security

Smart contracts protect several essential components of a multiparty business process. However, because the technology is so young, hackers are always finding new ways to jeopardize the goals of the companies that created the rules. Smart contract hackers were successful in stealing $50 million in cryptocurrency in the early days of Ethereum. Concerns about tool inconsistencies used to identify various smart contract security issues have also been raised by the IEEE.

* Integrity

One oracle (one of the streaming data sources that transmits event updates) must be protected from hackers who create fake events to cause smart contracts to run when they shouldn't. For complex circumstances, it must be configured to accurately generate events, which might be difficult.

* Alignment

Regardless of whether they are consistent with the intention and understanding of all parties, smart contracts can speed up the execution of activities involving numerous parties. However, this skill has the potential to amplify the harm that might result when things go awry, especially when there is no means to curtail or unwind irrational behavior. The research group Gartner has emphasized that this problem raises issues with smart contracts' manageability and scalability that have not yet been fully resolved.

* Management

The management and implementation of smart contracts are challenging. They are frequently set up in ways that make changes challenging or impossible. Although this can be seen as a security benefit, the parties are unable to modify the smart contract agreement or add new provisions without creating a new contract [7].

# Smart contract development platforms

There are five most popular smart contract platforms, Ethereum, Hyperledger Fabric, Corda, Stellar, and Rootstock.

* Ethereum

Ethereum is a decentralized platform that enables the execution of smart contracts. It translates smart contracts written in Solidity, Serpent, Low-level Lisp-like Language (LLL), and Mutan into machine code, which is then loaded and run by the Ethereum virtual machine (EVM). The computationally expensive proof-of-work (PoW) consensus technique is used by Ethereum. Instead of using Bitcoin (BTC), Ether (ETH) is used to compensate miners for the cost of fixing difficulties. To make up for the unstable value of ETH, the gas functions as an internal fee for completing a transaction. PoW is computationally intensive, yet it can be utilized for worthwhile tasks like solving mathematical problems and carrying out machine learning operations.

* Hyperledger Fabric

Smart contracts can be executed using the distributed ledger technology known as Hyperledger Fabric. Unlike Ethereum, which executes code in virtual machines (VMs), Hyperledger Fabric executes code in a Docker container. The project is managed by the Linux Foundation, with considerable funding provided by IBM. Instead of creating Ethereum smart contract languages, Fabric supports established high-level programming languages like Java and Go. It employs a key-value pair data model and is Turing complete.

The blockchain network of Fabric is permissioned, and before users can join the network, Certificate Authorities (CAs) must give their approval. Within the network of the permissioned blockchain, consensus can be reached fast.

* Corda

The distributed-ledger platform Corda is used to store and process historical data on digital assets. It is Turing incompletable and enables smart contracts. It is utilized to make it possible for private platforms to safely exchange digital assets. On private blockchain networks, consensus can be reached quickly thanks to Raft, a crash-fault-tolerant (CFT) ordering service that works on a "leader and follower" basis. A point-to-point messaging system is used by Corda in place of the global broadcasting found in blockchains.

* Stellar

Similar to Corda, Stellar is a dedicated platform for applications involving digital currencies. Compared to Ethereum, it is a more user-friendly and accessible cryptocurrency. Among the many languages it supports are Python, JavaScript, Golang, and PHP. A Stellar smart contract is a group of linked transactions that are carried out under different constraints and is created to send, store, and exchange value. The Stellar Consensus Protocol (SCP) is the organization's consensus method, while the account-based approach is the data model used by Stellar. Without depending on a closed system to monitor money transactions, SCP enables parties to reach an agreement.

* Rootstock

On top of Bitcoin, RSK is a public blockchain system that facilitates quicker transaction processing. It has Turing complete Rootstock smart contracts and is Ethereum compatible. Additionally, it contains a consensus mechanism based on PoW and an account-based data model. Although it is intended to support digital currency applications, it might put more stress on the Bitcoin blockchain. RSK must create a solution to this problem [8].

# Most common smart contract vulnerabilities

Knowing the most common blockchain smart contract vulnerabilities will help you better understand how to guard against exploitation. So let's examine the most typical smart contract flaws as well as actual examples of smart contract hacking.

* Reentrancy attack

One of the most famously exploitable flaws in smart contracts is reentrancy. It happens when a smart contract calls another smart contract in its code and then resumes execution after the new call has concluded. The vulnerable contract must make an external call in order to carry out this activity.

Scammers hijack these external calls and use the callback function to create a recursive call back to the contract. Using malicious code, they are able to construct a contract at an external address.

The withdraw function can be repeatedly used by the con artist to drain the contract funds when the smart contract forgets to update its status before sending funds.

* Front-running

Once submitted to the Ethereum network, smart contracts and transactions are publicly public, allowing block miners to choose the transactions with the greatest gas costs. Because of this visibility, malevolent parties are able to determine a smart contract's intended result before the transaction is recorded on the blockchain. Modern techniques like gas restriction and pre-commit mechanisms can assist secure a contract and provide protection against this. Pre-commit requires sending a hash in place of data in the first commit, whereas gas limiting entails only accepting transactions with a gas price below the designated threshold.

* Integer overflow and underflow

Solidity is one of many programming languages that share this smart contract vulnerability. By sending 1 Ether unit from a malicious address to the smart contract, a con artist can take advantage of it. The smart contract's balance will be compelled to cycle back to the highest value permitted (4.3 billion Ether) as a result. The actual result of the calculation and what was anticipated differ significantly due to both underflow and overflow difficulties, which results in the loss of the contract's cash. Use the 0.8 version of the Solidity compiler, which automatically tests for underflows and overflows, to prevent this hack.

* Simple logic error

One of the most frequent kinds of flaws in blockchain smart contracts is logic mistakes. These could include programming errors that are more significant and affect the security of smart contracts, typographical errors, and misinterpretations of specifications.

The good news is that during the smart contract audit, these issues can be found and fixed, hence it is advised that you do not skip this step before deploying your smart contracts on the blockchain.

* Block gas limit vulnerability

Blocks are kept from being too big thanks to the block gas limit. If a transaction uses too much gas, it won't fit in the block and won't be carried out.

As a result, if data is stored in arrays and accessed further through loops over these arrays, the transaction may run out of gas and receive a refund. A Denial of Service (DoS) attack might result from this.

* Default visibility

A function's visibility determines whether users can call it internally or outside. Functions have a public visibility state by default.

When authors of smart contracts fail to indicate which functions should be private or only accessible within the contract itself, this creates a problem.

* Timestamp dependence

Because of its decentralized nature, the block.timestamp function is not suggested for usage in smart contracts. It is advised to avoid utilizing it or allow a range of +900 seconds of error to mitigate this vulnerability. This will guarantee that the contract won't be significantly affected if the timestamp value is increased by a number between 1 and 900 seconds [9].

# Various steps for building a smart contract

1. Identify parties and establish the terms of the agreement.

The first step in constructing a smart contract is determining the parties involved and reaching a consensus on the contract's terms and conditions. This Agreement sets forth the provisions of the Agreement, the duties of the Parties, and the requirements for Contract Performance.

1. Define the conditions for contract execution

The requirements that must be satisfied in order for the contract to be performed are specified in the second stage. These terms are frequently stated as a list of requirements that must be met in order for the contract to be deemed enforceable.

1. Write the smart contract code

Writing the smart contract's code is the third phase. When the predetermined criteria are satisfied, the code will indicate the precise actions that must be taken to execute the contract.

1. Deploy the contract to a blockchain platform

The fourth step entails putting the smart contract into use on a blockchain network. In order to do this, the code must be uploaded to the blockchain network and its legitimacy verified.

1. Trigger the contract execution automatically

The fifth phase is the actual execution of the smart contract. The contract automatically executes and the blockchain network activates it when the preset conditions are met.

1. Record the contract details on the blockchain ledger

When a contract is performed, its data is added to the blockchain network. This includes the contract's terms, necessary conditions for execution, and the time and date of execution. Once they have been added to the blockchain ledger, the contract's specifics cannot be altered or deleted [10].

# Are smart contracts Enforceable?

Any findings about smart contracts must be tempered by the reality that states may adopt various viewpoints since the enforcement and interpretation of contracts in the United States are decided at the state level. Ancillary smart contracts can be used to satisfy the common law requirements of offer, acceptance, and consideration, which are generally used by state courts to establish enforceability. For instance, if a flight is delayed by more than two hours, an insurer may create a flight insurance product that automatically pays the insured.

The essential terms, like defining how the delay is determined, can be stated in a text-based contract, with an auxiliary smart contract handling the formation of the contract (payment of the premium) and execution (automatic payout following a verifiable delay). In 47 states, state legislation is based on the Uniform Electronic Transactions Act (UETA), which was first passed in 1999.

With a few exceptions, it states that electronic documents have the same legal weight as their written counterparts in terms of recordkeeping, signatures, and agents.

Even further, the UETA recognizes the legality of "electronic agents," which are computer programs or other automated tools used autonomously to start an activity or react to electronic records or performances in whole or in part, without approval or intervention from a person. This acknowledgement of smart contracts is foresighted. A contract or other record relating to a transaction may not be denied legal effect, validity, or enforceability simply because its formation, creation, or delivery involved the action of one or more electronic agents, according to the federal Electronic Signatures Recording Act (E-Sign Act), which recognizes the validity of electronic signatures and electronic records in interstate commerce.

Arizona and Nevada have updated their individual state versions of UETA to clearly include blockchains and smart contracts, indicating that there may be additional pressure to adopt unified definitions as more states follow their example. These amendments reflect the growth of blockchain and smart contract technology [11].

# Smart contracts can be created without coding

To make it simpler to become a blockchain developer, platforms for no-code or low-code smart contract development have been developed.

No-code and low-code platforms, however, have made it possible to almost immediately construct high-end smart contract apps. Almost anything can be built, and integrating low-code solutions with blockchain-powered apps is also made easier.

Because we can do more tasks in less time, this method of software development also boosts productivity. These products' manufacturers have already done the research for consumers. Various users can swiftly deploy their blockchain networks thanks to this.

The integration of multiple blockchain apps into a physical environment is another goal of low-code and no-code smart contract development [12].

# Applications of smart contracts

1. Real estate

Reduce the amount paid to the middleman and divide it among the participants. As an illustration, a smart contract to transfer apartment ownership whenever a specific number of resources have been sent to the seller's account.

1. Vehicle ownership

A blockchain can be used to implement a smart contract that records car ownership and maintenance. For instance, the smart contract may mandate car maintenance every six months, failing to which would result in the suspension of a driver's license.

1. Music industry

The ownership of music might be documented by the music business via a blockchain. When a music is utilized commercially, a smart contract that is integrated into the blockchain can be used to credit the owner's account with royalties. It can be effective in settling ownership conflicts as well.

1. Government elections

It would be very difficult to decrypt the voter address and amend the vote once the votes are stored in the blockchain, increasing the confidence against unethical acts.

1. Management

Many late or postponed choices can be streamlined and automated with the help of the blockchain application in management. Every choice is open to all parties with the power to influence it.

1. Healthcare

Smart contracts can be used to automate healthcare payment procedures to reduce fraud. Each procedure is recorded in the ledger, and the smart contract can ultimately calculate the total of all the transactions. Before the hospital bill has been paid and can be coded in the smart contract, the patient cannot be released from the facility [13].

# Smart contracts use cases

# Future of smart contracts

The concept of blockchain has transformed into a practical solution with applicability in actual corporate settings. To regulate transactional agreements housed on a blockchain network, smart contracts are the theoretical and technological rules and policies that are coded within the blockchain ecosystem. They are created as standalone programs that run on a blockchain network or as components of blockchain applications. A blockchain environment can be enabled or disabled via smart contracts, however this is problematic because it means high-level stakeholders and decision makers aren't involved. Smart contracts are quite sophisticated and require a lot of engineering knowledge to maintain.

When laying out their strategic IT plan, businesses must also take the benefits into consideration. Smart contracts are a particular kind of contract that use computer code to autonomously carry out a predetermined set of outcomes between two parties. This makes it less necessary for people to get involved in the contract execution procedure and speeds up the transaction compared to more conventional techniques. Additionally, it permits greater accuracy within the contract and better protection of the contracts themselves. Due to the immutability of smart contracts, which are constructed on a blockchain network, data cannot be lost, modified, or stolen without the appropriate authorizations. Smart contracts are being employed in a variety of sectors, including the utilities sector, where they effectively control how electricity is distributed in microgrids.

A microgrid's equipment are connected by means of smart sensors made possible by the internet of things (IoT). Microgrids are technological systems that track energy consumption, cut wasteful energy distribution, and create smart contracts based on actual consumer usage. Customers in a microgrid frequently purchase their energy requirements from a nearby prosumer through a digital transaction mediated by a smart contract. The transaction is frequently performed via the exchange of cryptocurrency. Due to the peer-to-peer network-based verification and real-time data reporting requirements, blockchain technology can dramatically reduce errors in REC distribution.

In addition to being effective, a smart contract-controlled microgrid protects the physical integrity of the grid by distributing power in accordance with actual usage. Traditional energy delivery cannot be based on precise consumption and runs the danger of causing a system overvoltage. Smart contracts enable contracts to be started more quickly and with fewer participants by using the public domain to manage an agreement between two parties [14].

# References

|  |  |
| --- | --- |
| [1] | "what is smart contractor," *https://www.techtarget.com/searchcio/definition/smart-contract.* |
| [2] | "benifits," *https://www.investopedia.com/terms/s/smart-contracts.asp.* |
| [3] | "disadvantages," *https://cystack.net/blog/pros-and-cons-of-smart-contracts.* |
| [4] | "history," *https://pontem.network/posts/the-history-of-smart-contracts.* |
| [5] | "uses," *https://www.gemini.com/cryptopedia/smart-contract-examples-smart-contract-use-cases#section-smart-contract-use-cases-in-finance.* |
| [6] | "pros and cons," *https://medium.com/coinmonks/smart-contracts-2022-introduction-945ff43162b2.* |
| [7] | "challenges," *https://www.techtarget.com/searchcio/definition/smart-contract.* |
| [8] | "platforms," *https://cointelegraph.com/learn/smart-contract-development-platforms#:~:text=Many%20of%20the%20existing%20blockchain,maturity%20in%20the%20growing%20community..* |
| [9] | "smart contract vulnerabilities," *https://pixelplex.io/blog/smart-contract-vulnerabilities/.* |
| [10] | "building steps," *https://cointelegraph.com/learn/what-are-smart-contracts-a-beginners-guide-to-automated-agreements.* |
| [11] | "enforceable," *https://corpgov.law.harvard.edu/2018/05/26/an-introduction-to-smart-contracts-and-their-potential-and-inherent-limitations/.* |
| [12] | "no coding," *https://betterprogramming.pub/3-no-code-platforms-to-build-and-deploy-smart-contracts-11a51196796a.* |
| [13] | "applications," *https://www.geeksforgeeks.org/smart-contracts-in-blockchain/.* |
| [14] | "future," *https://isg-one.com/articles/smart-contracts-the-future-of-contracting.* |