Hybrid Smart Contracts with Decentralized Oracles

Alexandru Gabriel ANDREI
National Institute for Research and Development in Informatics - ICI Bucharest
gabriel.andrei@ici.ro

Abstract: In the digital age, ensuring data workflow security is crucial. Blockchain's limitations in accessing external data have led to the development of Decentralized Oracles, facilitating the integration of real-world data with smart contracts. This article explores the challenges associated with incorporating external data into blockchain, evaluating various methodologies for achieving randomness. It assesses the security, decentralization and scalability of these approaches. Moreover, a Hybrid Smart Contract framework for NFT minting was developed, utilizing Verifiable Random Functions and Chainlink Oracles. This integration of Oracles and Hybrid Smart Contracts promises to revolutionize data security workflows within the decentralized landscape.

Keywords: Blockchain, Decentralized Oracles, Data Security, Hybrid Smart Contracts, Verifiable Random Functions.

INTRODUCTION

Nowadays, in the present modern world where everything is connected through technology, trust has become crucial and fragile at the same time. Trust forms the backbone of relationships, interactions, and progress. It's a binding force that holds communities, businesses and governments together. However, with the extend of the digital world, the notion of trust became even more fragile.

Now, a confluence of three powerful currents can be observed: a society struggling with eroding trust, an increasing digital world, and an arsenal of evolving technological applications. This convergence creates an unprecedented demand for trust, a unique kind that is tailored to this transformative epoch.

Progress propels society forward, introducing numerous changes and new paradigms. Cutting-edge technologies, both software and hardware, unveil new perspectives and digital spaces with new challenges.

In this context, decentralized technologies are here to stay in order to solve the problem of trust within this digital framework, by providing a fundamental layer of transparency, security, and immutability.

https://doi.org/10.54851/v6i1y202410

This article addresses a fundamental problem that blockchain technology poses and limits its core concept of decentralized and trustless system.

The core concept of blockchain was built around its capacity to offer a decentralized, transparent, and immutable ledger of transactions (Zheng et al., 2017). However, blockchain technology inherently lacks the capability to access data from external sources, which restricts its functionality and potential applications (Caldarelli, 2020). Integrating data from off-chain sources into the blockchain presents a significant data challenge, often accompanied by many security concerns. These security concerns could involve a wide range of issues, such as data manipulation or alteration, before it is recorded on the blockchain. Since blockchain transactions cannot be changed once they are recorded, any mistakes or malicious data could harm the entire blockchain network.

In addition, blockchain networks are decentralized, making it difficult to check the authenticity and accuracy of data obtained from external sources. Without a central authority to validate data authenticity, blockchain applications must rely on consensus mechanisms and cryptography to ensure the quality of off-chain data (Baboi, 2023).

Running smart contracts based on data from oracles adds a dependency on and trust in third-party services, which might not always be secure or reliable. This dependency on external data can create single points of failure and make blockchain applications more vulnerable to security threats.

Because of these issues, the so-called "oracles" have developed to address this limitation of blockchain. Oracles act as intermediaries that supply external data to blockchain smart contracts. They are key to enabling smart contracts to use real-world data, helping to automate operations and carry out complex conditional operations (Egberts, 2017).

To conclude, oracles connect the blockchain with the real world, enhancing the utility and flexibility of blockchain applications. Without oracles, blockchain applications would only rely on the data they can produce and verify on their own, weakening the basic idea of blockchain as a decentralized and trustless system (Ezzat et al., 2022).

BLOCKCHAINS ARE DETERMINISTIC SYSTEMS

In terms of computer science, blockchain is seen as a type of decentralized technology. It has transformed the manner in which data security and authenticity is perceived. Basically, blockchain is a technology that tracks information in an open and secure manner, without needing a central authority. This makes it very useful for many different industries.

By design, blockchain offers a strong data security structure. Each block holds a cryptographic hash of the preceding block, a timestamp, and transaction data, establishing its origin as the essential technology for a peer-to-peer electronic cash system (Nakamoto, 2008). This design makes it difficult to alter – once data is logged, changing or removing it is almost impossible.

This invaluable feature is a major benefit in keeping data secure. The decentralized nature of blockchain eliminates the need for a central authority, decreasing the risk of single points of failure. Each participant or node in the network holds a copy of the entire blockchain. So, even if one node is attacked, the data on the blockchain remains secure.

Moreover, all transactions are managed by a consensus mechanism. Generally, consensus mechanisms are protocols which ensure that all the nodes in a blockchain network agree on the legitimacy of transactions and on the state of the ledger (Salama et al., 2023). They help build trust among participants, without needing a central authority.

This complex mechanism within blockchain technology leads to the development of a deterministic system. This refers to the predictability and steadiness of outcomes

based on set rules and inputs. In blockchain, the deterministic aspect comes from the protocol's structure described earlier.

The deterministic nature of blockchain is evident in the transaction validation process. Each transaction follows preset rules programmed within the blockchain protocol. For example, the rules dictate how transactions are formed, verified, and added to the ledger. Once a transaction fulfills these conditions, it is considered valid and added to the blockchain. This deterministic validation process ensures transactions are handled consistently across the network, achieving consensus on the ledger's state.

Blocks are created through a process involving cryptographic hashing. The deterministic nature of hashing algorithms means they always give the same result for the same input, making them predictable. This ensures that creating blocks follows a set process, where the result depends solely on the input data and the rules in place.

Lastly, consensus mechanisms like Proof of Work (PoW) or Proof of Stake (PoS) contribute to the deterministic nature of blockchain systems (Bünz et al., 2017). These mechanisms set clear criteria for reaching agreement among network participants on the legitimacy of transactions and the ledger state. Through the deterministic execution of these consensus algorithms, nodes in the network agree on a single version of the blockchain.

On the one hand, this deterministic feature within blockchain technology ensures predictability, consistency, and reliability. On the other hand, this pattern involves some downsides, particularly in its limitation to access real-time data from external sources.

CASE STUDY - ACHIEVING RANDOMNESS IN BLOCKCHAIN

In the field of blockchain technology, getting randomness right is very important for various applications. However, because blockchains are by deterministic nature, as it has been earlier mentioned, making truly random numbers is really a challenge (Werapun et al., 2023). Blockchains depend on consensus mechanisms to agree on the network's status. Introducing randomness might compromise these consensus mechanisms, because random results vary, which could affect the process of achieving consistent agreement throughout the network. Unlike deterministic algorithms that give the same result with the same input every time, randomness goes against this predictability.

In this case study, different approaches were explored to create randomness using three methods: "Randomness from the block hash", "Randomness from on-chain collaboration", and "Randomness from oracles".

Consider a decentralized application that works on a blockchain. Users buy tickets to join a lottery, and winners are picked randomly. Three smart contract architecture solutions are analyzed in the following:

Randomness from the Block Hash

This methodology involves getting the hash of the previous block using the "blockhash" function available in Solidity. The output of this function is a bytes32 hexadecimal value representing the block hash. To convert this hexadecimal value into a usable format, it is transformed into a uint256. The process involves extracting the hash from the previous block:

```
function generateRandomNumber() public view returns (uint256) {
    // Get the block hash of the previous block
    bytes32 previousBlockHash = blockhash(block.number - 1);

    // Convert the bytes32 hash into a uint256 value
    uint256 randomNumber = uint256(previousBlockHash);

    return randomNumber;
}
```

A big issue here is that the timestamp and block hash can be manipulated by miners. Malicious players could exploit this method by manipulating the block timestamp or trying to control the mining process to influence the outcome of random number generation. For instance, they could try to mine blocks at specific times to influence the block timestamp in their favor, thus biasing the outcome of the random number generation and increasing their chances of winning the lottery unfairly.

Randomness from On-chain Collaboration

In this approach, randomness is generated through the collaboration of multiple participants or entities within the blockchain network. Smart contracts set up a process where various nodes or participants make inputs, which are then merged to create a random result. This collective action adds extra security and decentralization to the randomness generation, lowering the risk of manipulation or bias by any one party. The individuals produce random numbers, as mentioned earlier, using the block hash. Then, all the inputs are put together into a single data structure. This merged data is hashed using a cryptographic hashing algorithm like SHA-256. The hash value is turned into a numerical value using modulo arithmetic. This method improves proportionally with more participants. However, the system could still be at risk, if a group of participants conspires together. If they work together, they could impact the randomness generation, by coordinating their inputs or influencing the hashing process.

Additionally, as more participants join, the process of gathering, merging, and hashing their inputs becomes more resource-intensive and less efficient. This can create scalability issues, especially in networks with many participants or high transaction volumes.

Randomness from Oracles

This method involves utilizing oracles, which are trusted third-party services that supply external data to smart contracts on the blockchain. A blockchain oracle refers to any device that interacts with the real world to provide external data to smart contracts. However, using a centralized oracle introduces a point of failure that is incompatible with the principles of a decentralized network. Instead, a decentralized network of oracles can be used to bring data into smart contracts. This network of oracles obtains data from providers, sends it through a network of oracles which then uses a median to find the correct value, and delivers it in one transaction to a reference contract that exists on the chain. This concept leads to hybrid smart contracts, which mix on-chain data with off-chain data. Additionally, the reference hybrid smart contract provides input for other smart contracts within the blockchain network, as shown in Figure 1.

Achieving randomness in blockchain through oracles also involves using Verifiable Random Functions (VRFs). A Verifiable Random Function (VRF) is a cryptographic function that takes an input and produces a random output unique to that input. It also creates a proof that can demonstrate the output's authenticity without revealing extra information. This allows the output to be verified while keeping the generation process secret. Therefore, in addition to the random number, the VRF also outputs a proof that anyone can check to confirm the random number's correctness without knowing the private key.

VRFs were first proposed by Silvio Micali, Michael Rabin, and Salil Vadhan (1999) to fulfill the need for a function that could generate pseudorandom outputs that are publicly verifiable.

Chainlink, a decentralized oracle network, has added VRF functionality to its setup. Chainlink's VRF operates within a decentralized oracle network, ensuring that random numbers are generated in a tamper-resistant and transparent manner. By distributing the generation process across many nodes, Chainlink's VRF enhances security and reliability (Chainlink, 2020).

In the context of Chainlink's Verifiable Random Function (VRF), the Chainlink VRF coordinator contract is considered the reference contract, as shown in Figure 1. This contract plays a central role in coordinating randomness requests and generating verifiable random numbers.

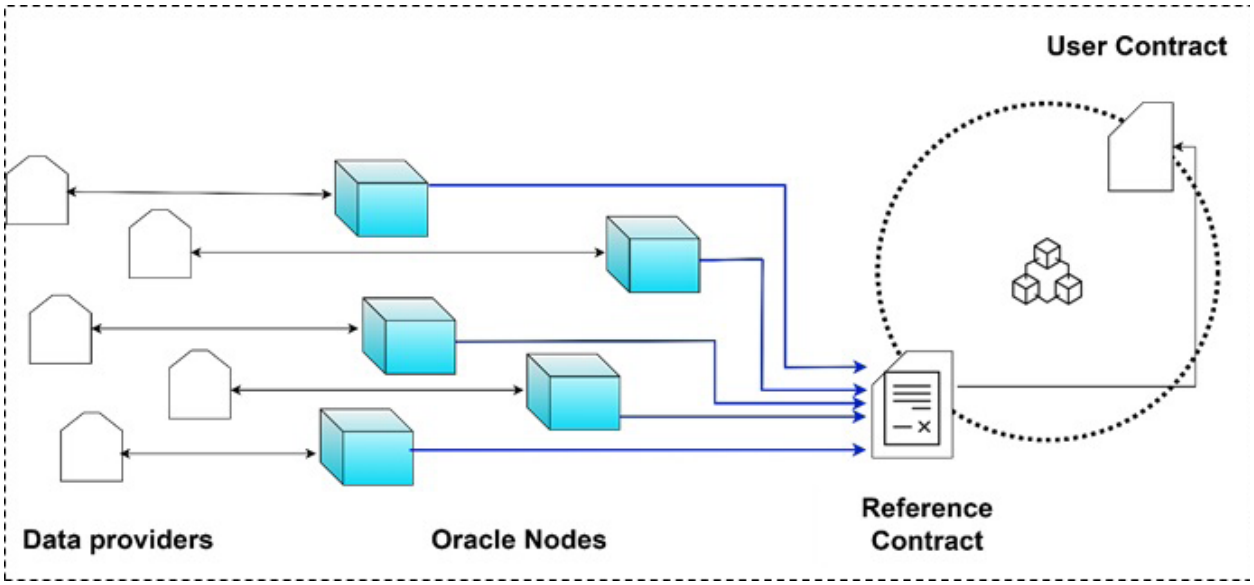

Figure 1. *General data workflow for oracle networks*

In order to generate a random number, the Chainlink infrastructure was involved. The methodology for generating randomness with oracles can be emphasized in the following steps (Chainlink, 2023):

- Coordination of Randomness Requests: The Chainlink VRF coordinator contract receives randomness requests from smart contracts deployed on the blockchain. These requests typically include parameters such as the key hash, fee, and user-provided seed;
- Random Number Generation: When a request for randomness is received, the Chainlink VRF coordinator uses its VRF abilities to create a random number. This process ensures that the generated random number is both unpredictable and verifiable;
- Verifiable Proofs: In addition to the random number, the Chainlink VRF coordinator creates cryptographic evidence that confirms the accuracy and authenticity of the generated randomness. These pieces of evidence allow anyone to check if the random number is valid, without needing to trust anyone else;
- Fulfillment of Randomness Requests: After creating the random number and its proofs, the Chainlink VRF coordinator completes the randomness request, by sending both the random number and proofs back to the smart contract that made the request;
- Integration with Oracle Nodes: The Chainlink VRF coordinator works with a network of decentralized oracle nodes to gather randomness and maintain the trustworthiness of the randomness generation. These oracle nodes contribute to the decentralization and security of the VRF mechanism.

Continuing to explore the issue addressed by this article, a comparative analysis of the three presented methods was conducted in order to assess their respective levels of security, decentralization, resistance to manipulation, scalability and transparency in relation to one another, as shown in Table 1.

Table 1. *A comparative analysis of the three presented methods*

Criteria	Block Hash	On-Chain Collaboration	Oracles
Security	Vulnerable to manipulation of block timestamp and hash by miners	Vulnerable to collusion among participants	Relies on trusted third-party services or decentralized network of oracles
Decentralization	Relies on blockchain's decentralized nature but susceptible to manipulation	Adds decentralization through collaboration but still susceptible to collusion	Can be decentralized if using a network of oracles, otherwise relies on trusted third parties
Resistance to Manipulation	Vulnerable to manipulation by miners	Vulnerable to collusion among participants	Resistance depends on the trustworthiness and decentralization of oracles
Scalability	Efficient, minimal resource usage	Resource-intensive as participants increase	Depends on the efficiency and scalability of the oracle network
Transparency	Transparent within the blockchain	Transparent if properly implemented and audited	Depends on the transparency and auditability of the oracle network

In conclusion, while each method of achieving randomness in blockchain has its own strengths and limitations, depending on the specific requirements and context of the application, the utilization of oracles and hybrid smart contracts provides the best solution for achieving truly randomness.

BUILDING A HYBRID SMART CONTRACT FRAMEWORK FOR MINTING NFTS

Randomness is crucial for Non-Fungible Tokens (NFTs), as it ensures their uniqueness, variety and dynamic content. In response to the growing demand for innovative NFT solutions, a framework for minting NFTs was developed based on hybrid smart contract and advanced features such as Verifiable Random Functions, as it can be observed in Figure 2.

In the most common application, the metadata for NFTs, which includes the corresponding images, is stored in InterPlanetary File System (IPFS). This approach presents some drawbacks concerning data security and immutability. Storing data on-chain offers superior security compared to storing it in IPFS, due to benefits from the inherent security features of blockchain technology. IPFS relies on distributed storage which may introduce some vulnerabilities. As a result, the presented architecture also introduces on-chain SVG images storage.

The developed smart contract was designed to orchestrate the creation of an NFT with random traits. It was built on the Ethereum blockchain and uses features like ERC721 compliance, Chainlink VRF integration, and on-chain SVG image storage.

Therefore, the framework called "RandomNFT" involves generating multiple types of NFTs with

various features, randomly selected from a predefined array of changes set by the contract's owner. In this example, three categories were chosen, based on an array of changes with three elements set at 10%, 30% and 60%.

The constructor function of the smart contract accepts several parameters, including the address of the VRF, a subscription ID, a key hash representing the gas lane, a minting fee, a gas limit for callbacks, and an array of token URIs. These parameters are essential components contributing to the embedded VRF capabilities within the contract. During deployment, the constructor function initializes the contract by assigning values to each parameter, enabling the seamless integration of VRF functionalities. Furthermore, it initializes the array of token URIs, which serves as metadata pointing to SVG images associated with each category of NFTs. This ensures that the contract adheres to the ERC721 standard, while providing comprehensive metadata for each category of NFTs, thereby enhancing compatibility and interoperability within the NFT ecosystem.

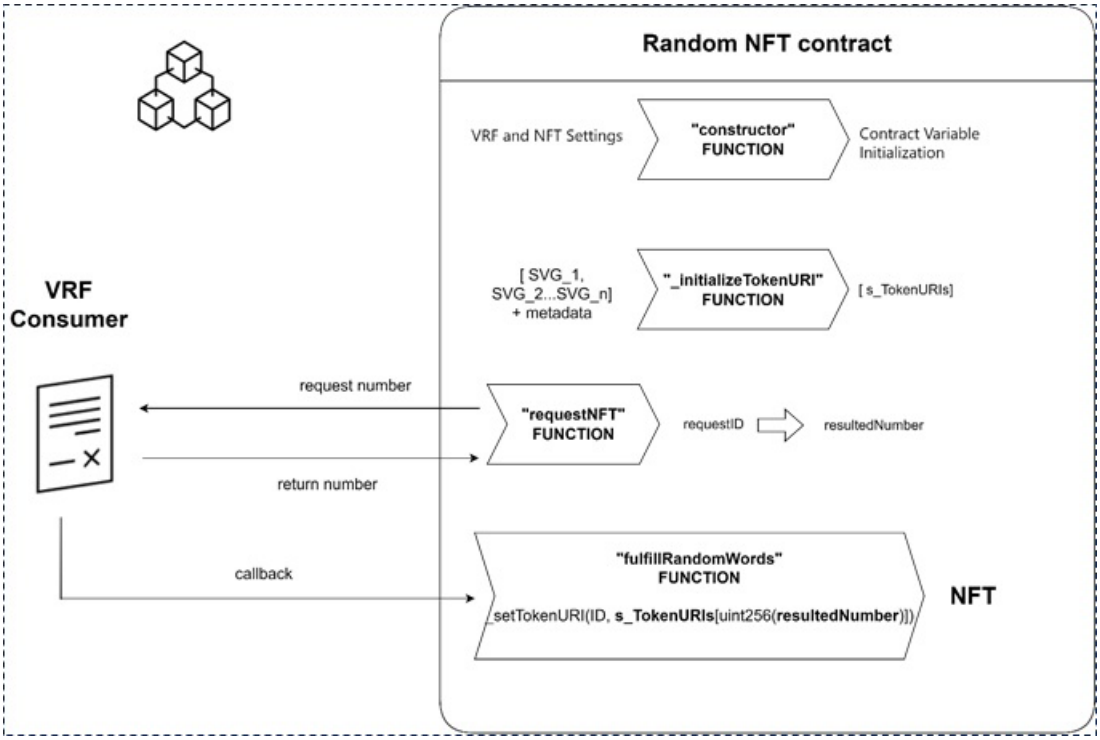

Figure 2. *Smart Contract Architecture*

The contract inherits a reference contract called "VRFConsumerBase" provided by Chainlink to facilitate integration with their VRF service. This contract acts as an intermediary between the smart contract (in this case, "RandomNFT") and the Chainlink VRF coordinator. It provides functions and mechanisms for interacting with the VRF service, such as requesting random numbers and handling their fulfillment. By inheriting from "VRFConsumerBase", the "RandomNFT" contract gains access to these functionalities, allowing it to leverage Chainlink's VRF service to generate verifiable randomness for minting dynamic NFTs.

Dynamic Non-Fungible Tokens (NFTs) represent an evolution of static traditional NFTs. Unlike static NFTs, which represent fixed digital assets, dynamic NFTs possess the capability to change, adapt, and interact with external environments over time.

The concept of dynamic NFTs introduces a new dimension of interactivity, versatility, and utility to the NFT ecosystem. These tokens can

incorporate real-time data feeds or external triggers to modify their properties, behaviors, or appearances dynamically. This dynamic behavior opens countless possibilities.

During the contract initialization, the "_initializeTokenURI" function is invoked to handle the array of SVG image URIs provided during deployment. This function iterates over each URI, converting it into a data URI format through the "svgToImageURI" function. Here, the SVG content is encoded into a data URI format using Base64 encoding:

```
string memory svgBase64Encoded = Base64.encode(
      bytes(string(abi.encodePacked(svg)))
  );
  return string(abi.encodePacked(baseURL, svgBase64Encoded));
```

The resulting data URI, now containing the SVG image in Base64 format, is then utilized within the "formatTokenURI" function to construct a complete token URI. Within "formatTokenURI", the function assembles a comprehensive description incorporating essential metadata such as the name, description, and attributes relevant to the NFT. This process makes sure that every token URI contains all the important information needed to describe the NFT, which helps define what the NFT is like and what makes it unique. Further, these formatted token URIs are stored in the "s_tokenUris" array, enabling the contract to associate specific SVG images with the NFT categories generated later during the minting process.

When a user requests the creation of an NFT by calling the "requestNft" function, a request for random number is made to the Chainlink VRF, associating the request with the sender's address. Once the VRF process generates the random number, an internal callback is triggered, invoking the "fulfillRandomWords" function. Within the "fulfillRandomWords" function, the random number generated by the Chainlink VRF is processed to determine the category of the future NFT. This processing involves applying modulo arithmetic to the random number which leads to a number between 0 and 99.

Once the random number is transformed, it is utilized to map the NFT to one of the predefined categories. These categories are established based on predefined chance arrays, each representing a distinct value situated within the range [0-99] . Through a comparison between the transformed random number and the chance arrays, the contract determines the category to which the NFT belongs.

To connect the NFT with its visual representation, the contract retrieves a token URI from the "s_tokenUris" array. This ensures that each NFT is aligned with a unique SVG image representation based on its category. The token URI encompasses metadata and a link to the SVG image, providing a comprehensive description of the NFT.

Overall, the architecture of the presented contract embodies transparency, randomness, and efficiency in generating NFTs, making it a reliable and versatile solution for NFT creation on the Ethereum blockchain.

The presented framework could be implemented in a real-world scenario, like a limited collection of assets, each associated with an NFT containing random digital traits. In this case, every asset has a unique identification and is represented by an NFT, which includes different digital characteristics and levels of rarity. By using this concept, stakeholders can utilize blockchain technology and NFTs to improve customer involvement and loyalty to the brand. The transparent and decentralized characteristics of blockchain technology maintain the integrity and authenticity of each NFT, giving customers trust in the ownership and history of their digital assets.

The utility of hybrid smart contracts is not limited to the NFT ecosystem, as they also have the potential to improve industries such as supply chain, finance, healthcare, and any other industry where a set of rules based on real-world information can be encoded. In the

case of supply chain management, hybrid smart contracts will be the backbone for integrating off-chain real-time data, such as temperature or humidity, from sensors during shipping situations.

In finance, hybrid smart contracts could facilitate complex blockchain transactions. These contracts can automate various financial processes based on a set of predefined rules linked to real-time market data, which are off-chain information.

In healthcare, if we imagine a blockchain ecosystem that records patient data, hybrid smart contracts can automate administrative tasks such as insurance claims processing based on predefined rules and conditions.

CONCLUSION

Decentralized oracles are an important mechanism for decentralized applications, and, without them, blockchain networks would be irrelevant and incomplete systems. They work as a bridge between blockchains and outside data, enhancing smart contracts, by allowing them to use information from the real world.

This research demonstrates the critical role of decentralized oracles in improving blockchain protocols by integrating external data into smart contracts. According to the presented methodology, achieving randomness in a deterministic system – a core functionality for many applications – can be challenging. Therefore, using oracles is the best solution for this purpose. One way to bring off-chain data to the blockchain is through VRF from the Chainlink network. As a result, the concept of hybrid smart contracts was introduced and a framework to get off-chain data, like random numbers, in this case, was developed. The developed framework can be tailored to any application that needs randomness by collaborating with external oracle networks. This collaboration creates hybrid smart contracts that can change and adapt to real-time events, creating numerous new opportunities for decentralized applications.

Looking at the bigger picture, technology initiates a chain reaction, where building one application layer leads to changes in others. This process is a fundamental part of technological growth. As one area improves, it helps other areas grow too, with effects extending far beyond the initial applications, shaping technological development and impacting different parts of society.

Blockchain technology is known for being open, decentralized, and immutable, serving as the foundational layer for many applications and ecosystems. As blockchain technology evolves, it prepares for new kinds of innovation like decentralized oracles and dynamic systems. The utility of oracles and hybrid smart contracts extends beyond achieving randomness generation. It represents a fundamental evolution in the capabilities of blockchain technology. Finance, supply chain, and even healthcare can be improved with various kinds of decentralized applications, now possible due to the utility of hybrid smart contracts.

REFERENCE LIST

Baboi, M. (2023) Security of Consensus Mechanisms in Blockchain. *Romanian Cyber Security Journal*. 5(2), 45-53. doi: 10.54851/v5i2y202305.

Bünz, B., Goldfeder, S. & Bonneau, J. (2017) Proofs-of-delay and randomness beacons in Ethereum. In: *Proceedings of the Crypto Economics Security Conference (CESC), 2–3 October 2017, Berkeley, CA, USA*, pp. 1–11.

Caldarelli, G. (2020) Understanding the Blockchain Oracle Problem: A Call for Action. *Information*. 11(11), 509. doi: 10.3390/info11110509.

Chainlink. (12 May 2020) *Chainlink VRF: On-Chain Verifiable Randomness.* https://blog.chain.link/chainlink-vrf-on-chain-verifiable-randomness/ [Accessed 15th February 2024].

Chainlink. (2023) *VRF in Blockchain.* https://docs.chain.link/vrf/v2/subscription/examples/get-a-random-number/ [Accessed 15th February 2024].

Egberts, A. (2017) The Oracle Problem - An Analysis of how Blockchain Oracles Undermine the Advantages of Decentralized Ledger Systems. *SSRN Electronic Journal*. doi: 10.2139/ssrn.3382343.

Ezzat, S. K., Saleh, Y. N. M. & Abdel-Hamid, A. A. (2022) Blockchain Oracles: State-of-The-Art and Research Directions. *IEEE Access.* 10, 67551-67572. doi: 10.1109/ACCESS.2022.3184726.

Micali, S., Rabin, M. & Vadhan, S. (1999) Verifiable random functions. In: *Proceedings of the 40th Annual Symposium on the Foundations of Computer Science (FOCS `99), 17– 18 October 1999, New York City, NY, USA*. Washington, DC, USA, IEEE Computer Society Press. pp. 120-130.

Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. *Bitcoin*. https://bitcoin.org/en/bitcoin-paper [Accessed 24th February 2023].

Salama, M., Hussein, Z. & Hassan, S. (2023) Evolution of blockchain consensus algorithms: a review on the latest milestones of blockchain consensus algorithms. *Cybersecurity*. 30 (6). doi: 10.1186/s42400-023-00163-y.

Werapun, W., Karode, T., Suaboot, J., Arpornthip, T. & Sangiamkul, E. (2023) Native VRF: A Simplified Decentralized Random Number Generator on EVM Blockchains. *Systems.* 11(7), 326. doi: 10.3390/systems11070326.

Zheng, Z., Xie, S., Dai, H., Chen, X. & Wang H. (2017) An overview of blockchain technology: Architecture, consensus, and future trends. In: *6th IEEE International Congress on Big Data, 25-30 June 2017, Honolulu, HI, USA. New Jersey, USA*, The Institute of Electrical and Electronics Engineers (IEEE). pp. 557-564. doi: 10.1109/BigDataCongress.2017.85.