NFSU JOURNAL OF CYBER SECURITY & DIGITAL FORENSICS

NFSU – Journal of Cyber Security and Digital Forensics
Special Issue – 1, 2024
E – ISSN – 2583-7559

A Secure Smart Contract Development in Blockchain: Attacks and Defense

Digvijaysinh Rathod
School of Cyber Security and Digital Forensics,
National Forensic Sciences University
Gandhinagar, Gujarat (India)

Dr. Mukti Padhya
School of Cyber Security and Digital Forensics,
National Forensic Sciences University
Gandhinagar, Gujarat (India)

Naveen Kumar Chaudhry
School of Cyber Security and Digital Forensics,
National Forensic Sciences University
Gandhinagar, Gujarat (India)

Sarang Rajvansh
School of Cyber Security and Digital Forensics,
National Forensic Sciences University
Gandhinagar, Gujarat (India)

Ramya Shah
School of Cyber Security and Digital Forensics,
National Forensic Sciences University
Gandhinagar, Gujarat (India)

Param Ahir
School of Cyber Security and Digital Forensics,
National Forensic Sciences University
Gandhinagar, Gujarat (India)

***Abstract*— A blockchain is public ledger that stores all finalised transactions in a sequence of blocks. The length of this chain steadily increases as new blocks are consistently added to it. The blockchain was originally employed for decentralised Bitcoin transactions, but it has since acquired wider applications. Blockchain technologies are crucial in implementing the trust system suggested by smart contracts. A smart contract is a software programme that operates on the blockchain and is ensured to execute accurately as a result of the consensus protocol. Through the utilisation of asymmetric cryptography and distributed consensus algorithms, we have effectively guaranteed user security and upheld the consistency of the ledger. Our literary analysis indicates that smart contracts are susceptible to a range of attacks and security concerns, allowing an external entity to manipulate the smart contract. This research paper examines resource-exhaustion attacks, transaction-ordering dependence attacks, exception disorder attacks, and parity wallet hacks associated with smart contract development. In addition, we propose a methodical strategy to tackle these attacks and enhance the security of smart contracts..**

Keywords—blockchain, smart contract, Go-ethereum, Homebrew, and Solidity

I. INTRODUCTION

The blockchain is a self-governing database [6, 15] that maintains records or a public ledger of all finalized transactions or digital events, which are distributed among participating entities [1, 6]. The legitimacy of every transaction documented in the public ledger is established through the consensus of the majority of participants in the system. Once a transaction is recorded, it becomes immutable and cannot be erased, ensuring the blockchain maintains a permanent and dynamic ledger of all past transactions [6]. This immutable record facilitates decentralized transactions [10].

Structurally, a blockchain is a hierarchical arrangement of blocks, each positioned on top of the preceding one, with the block at the bottom serving as the base of the stack [11]. Each block is interconnected with the others and recognized as the preceding block in the chain. The hash of each block is computed using the SHA-256 secure hash algorithm applied to the block's header. The salient characteristics of blockchain technology include:

- Independent of centralised technologies: In any typical transaction between two entities, the verification of information transactions requires the participation of a third party [12]. Blockchain technology enables the decentralised exchange of information, eliminating the necessity for centralised entities. In contrast, blockchain technology independently creates and maintains a decentralised network. A blockchain network comprises multiple individual computers that supplant large, centralised

entities, epitomising the most pristine manifestation of such a network.

- Incentivized distributed network: Miners are compelled to participate in the network because of the incentives provided in the form of crypto-currency rewards. Each computer receives a small amount of cryptocurrency from the blockchain when it is selected to transmit information as a block [7].
- Cryptography-based distributed network: The blockchain transaction is immutable due to the utilisation of cryptography for transaction security [20]. Each computer in the network strives to solve a cryptographic puzzle and vies for recognition as the next block used in the transaction. The blockchain additionally produces an unchangeable record of all previous transactions, thus creating a ledger of transactions. Any person can analyse a public blockchain to understand the chronological data it holds. When another person examines the ledger at a different time or place, they will see an identical copy of the transaction history. The record is immutable as it is stored within the blockchain code and cannot be altered by any central authority [7].

II. VARIOUS TYPES OF BLOCKCHAIN

A. *Public blockchain*

Any individual can participate in a public blockchain by downloading the necessary software on their device. They can then store, send, and receive data. [9,21] It exhibits complete decentralisation. Due to the extensive distribution of records among numerous participants, tampering with transactions in a public blockchain is extremely unlikely [9]. The propagation of transactions and blocks is time-consuming due to the extensive number of nodes involved [9].

B. *Private blockchain*

This type of blockchain is under the complete control of a single organisation, which has the authority to determine the ultimate consensus [9,21]. It demonstrates complete centralization as it is governed by a solitary entity [9]. Transactions in a private blockchain are susceptible to tampering due to the limited number of participants involved [9]. In a consortium blockchain, the validation of blocks is carried out by a specific group of nodes that have been chosen for this purpose. By reducing the number of valuations, consortium blockchain and private blockchain can achieve higher efficiency [9]. Participation in the consensus process is restricted to nodes with permission [9].

III. NEED FOR BLOCKCHAIN

Blockchain fundamentally relies on a decentralised, digitalized, and distributed ledger model. Due to its inherent characteristics, this approach is more resilient and resistant to security breaches compared to the exclusive, centralised models presently employed in the trade ecosystem. The utilisation of blockchain provides a significantly superior method for establishing and verifying identity compared to current systems. Blockchain technology significantly streamlines the direct transfer of trade assets and enhances trust. This is accomplished by offering distinct, unforgeable identities for assets, coupled with an immutable record of their ownership [11].

The utilisation of blockchain as a ledger system ensured the secure recording and facilitation of transactions for this new currency, without the need for oversight or regulation from banks or governments. Bitcoin can be regarded as the initial application that utilises blockchain technology [14].

A. *How does blockchain enable transactions?*

To conduct digital transactions, you are assigned a distinct identity through a set of cryptographic keys. The private key and public key [20] constitute your keys, which are used in combination to generate a digital signature. The public key serves as a means for others to authenticate your identity. By utilising your private key in conjunction with your public key, you possess the ability to digitally sign and validate various actions on behalf of your digital identity. In the realm of cryptocurrencies, your wallet address (public key) serves as an identifier, while your private key grants you the authority to execute transactions, withdrawals, and other operations involving your digital assets, such as cryptocurrencies. Each transaction is consistently authenticated by the individual responsible for its authorization. The transaction entails Alice sending 0.4 BTC to Bob, including Bob's address (public key), and being authenticated by a digital signature utilising both Alice's public and private keys. This transaction is recorded in the ledger of the blockchain, indicating that Alice transferred 0.4 BTC to Bob. The transaction is accompanied by a timestamp and a unique identification number. Upon the occurrence of this transaction, it is disseminated to a decentralised network [14] of nodes, which are essentially other digital entities that confirm the transaction and include it in the ledger.

Every transaction recorded in the ledger will possess distinct attributes, including a digital signature, a public key, a timestamp, and a unique identifier. Every transaction will be linked, meaning that if you go back one transaction in the ledger, you will find evidence of Chuck sending Alice 0.8 BTC at a specific point in time. Upon reviewing previous transactions, it is possible to observe that Dan had previously sent Chuck 0.2 BTC at an earlier point in time. The anonymity of cryptocurrencies stems from the utilisation of randomised alphanumeric sequences as public keys, which eliminates the need for individuals to sign transactions using their actual names or aliases. A public key does not provide information about the true identity of the individual associated with it.

Smart contracts play a pivotal role in the realm of digital currencies, serving as the backbone for ensuring transactional integrity within blockchain networks [16, 17]. Ethereum stands out as a leading framework for the deployment of these contracts, crafted with an intricate execution architecture in mind. Solidity, a language specifically forged for scripting on Ethereum, caters to the unique needs of contract development. Each smart contract is uniquely identified by a 160-bit address, with its script securely ensconced within the blockchain [1]. In prominent digital currencies, the activation of smart contracts is triggered by directing transactions towards the contract's unique address [1]. When the blockchain confirms a transaction directed at a contract address, it prompts the entire mining consortium to execute the contract's script, drawing upon the latest blockchain state and transaction details as input [1]. The network then collectively reaches a consensus on the outcome and the updated state of the contract through a concerted consensus mechanism [1]. Ethereum, known for its comprehensive computational capabilities, is a standout platform for smart contracts. Unlike Bitcoin, which is more restrictive, Ethereum supports dynamic contracts that allow for the recording and subsequent access of data on the blockchain in various contexts [1]. This flexibility marks a significant advancement in the capabilities of blockchain technology.

IV. Introduction to Smart Contracts

A contract is a clear and explicit agreement or a conventional means to formalise a connection or relationship. Usually, they are utilised in business transactions, although they can also include other personal connections such as politics, marriage, and more. The concept of smart contracts was first introduced by Nick Szabo. Smart contracts involve the integration of different types of contractual clauses, such as liens, bonding, and property rights, into the hardware and software that we utilise. The goal is to create a situation where it is expensive, and in certain instances, even unfeasible, for parties to violate the contract. A tangible illustration of a rudimentary forerunner to smart contracts is the ubiquitous vending machine. The machine functions within a pre-established limit of potential loss, guaranteeing that the amount of money in the cash register is always lower than the expense of violating the mechanism. The machine accepts coins and employs a simple mechanism, which poses a fundamental design challenge related to finite automata, to distribute change and products in an equitable manner [2]. Smart contracts aim to integrate contracts into digitally controlled assets, going beyond just vending machines. These contracts are referred to in a dynamic and rigorously enforced manner, providing improved monitoring and validation. While vending machines and email use an asynchronous protocol to communicate between the company and the customer, certain smart contracts involve multiple synchronous steps that require the participation of multiple parties. The user's input is "[2]". Smart contracts are automated contracts where the contractual terms are encoded directly into computer code, allowing them to execute themselves. They are mechanised and enforceable, obviating the necessity for intermediaries and ensuring transparency and security. In essence, it is a contract that is completely sealed and cannot be altered or interfered with. A smart contract is a computerised agreement that regulates the management of users' digital assets, outlining the privileges and responsibilities of the participants, and is autonomously executed by a computer system. It serves as both a computer procedure and a participant in a contract. The system reacts to incoming messages, retains data, and possesses the capability to transmit messages or values to external entities. A Smart Contract is a programmable entity that can be trusted to temporarily store assets and carry out predetermined instructions [3]. A smart contract is a software programme that runs on the blockchain and is guaranteed to execute accurately due to the consensus protocol. A smart contract programme comprises a function, a storage component, and a defined amount of cryptocurrency. Users engage with the contract by sending transactions to the public address of the contract on the blockchain. The contract code is immutable and unmodifiable once it has been written. Because they operate on an open network, they are vulnerable to attacks.

V. Literature Survey

The blockchain is one of the six layers that make up the entire system. These layers consist of the network, transaction, blockchain, trust, application, and security layers. Each layer in blockchain technology holds significance, but the security layer is particularly crucial. Blockchain technology is susceptible to various forms of attacks, including eclipse attacks and selfish mining. Lim et al. [22] conducted a security analysis and demonstrated security vulnerabilities, such as DDOS and 51% attacks. The authors Vasek et al. [26] Provides evidence regarding Distributed Denial of Service (DDoS) attacks. The study conducted by Luu et al. The paper [24] introduces a security attack known as the verifier's dilemma. Eyal and Sier [27] Present a Selfish Mine attack and the unauthorised access of private accounts. Zyskind et al [23] introduced the blockchain architecture as a means of safeguarding personal data. Armknecht and colleagues.To ensure security and privacy in the Ripple system, the following measures should be implemented: Decker & Wattenhoffer The study conducted by [29] focused on the concept of transaction malleability, as explored by Andrychowicz et al. A malleability attack has been executed, and its consequences have been assessed. Bos et al. [31] have demonstrated that elliptic curve cryptography (ECC), which is utilised in Bitcoin, lacks the necessary level of randomness.

The literature shows that authors focus on DDOS, eclipse, selfish mining, verifier's dilemma but the author focused on resource-exhaustion attacks; transaction-ordering dependence attack, exception disorder attack, and parity wallet hack related to smart contract development. The author also proposed a methodology to defend against such attacks.

VI. TYPES OF ATTACKS

This section will explore the various types of vulnerability linked to smart contracts, as well as potential attack scenarios.

A. Resource-exhaustion attacks

In Ethereum, miners are rewarded with a fee for their computational effort in order to ensure fair compensation. For instance, as demonstrated in the code below, the individual who successfully solves the puzzle is granted the prize. Ethereum has a predetermined quantity of currency known as "gasLimit." When a user activates a contract, they must specify the gasLimit for execution, as well as the gasPrice for each unit of gas. The miner who successfully mines a contract receives the transaction fee, which is calculated by multiplying the gasLimit by the gasPrice. If an execution surpasses the gasLimit, it will be abruptly halted and an exception will be thrown. The state will then be rolled back to its initial state, as if the execution never occurred. [1].

B. Transaction-ordering dependence attack

The source code - 1 reveals that the malicious owner can exploit this transaction-ordering dependence to obtain a significant financial advantage. There is a 12-second delay between the broadcasting of a transaction and its inclusion in a block. During this time frame, the malevolent proprietor has the ability to eavesdrop on the network in order to determine if anyone has submitted their solution to his contract. If the answer is affirmative, he proceeds to adjust his reward, reducing it to nearly zero. There is a possibility that both transactions are included in the new block. In this scenario, when the user's transaction is executed before the other transaction, the user benefits from a free solution without taking any action, while the other person's effort goes to waste and they receive no credit [1].

Source code - 1

```
contract EntropySource {
    uint private accumulatedFunds = 0;
    // Utilizes the current block's timestamp as a foundational random element
    uint256 baseSeed = block.timestamp;

    // Generates a pseudo-random number
    function generateRandomNumber() public returns (uint256 randomNumber) {
        uint256 dynamicModifier = baseSeed * block.number / (baseSeed % 5);
        uint256 enhancedSeed = block.number / 3 + (baseSeed % 300) + accumulatedFunds + dynamicModifier;
        // Utilizes the hash of a specific past block as a random factor
        uint256 calculatedHash = uint256(blockhash(enhancedSeed));
        // Ensures the output is within a specified range
        return (calculatedHash % 100) + 1;
    }
}
```

The above code is used to decide the outcome of a jackpot. It stores L_Pay and block's timestamp as variables. In the random function in line 4, a random number is generated using the previous block's number and the current system's timestamp as seed.

C. Occurrence of an exceptional disorder attack

As demonstrated in the source code - 2, we will illustrate this issue using an example - the KingOfEtherThrone (KoeT). This contract grants the individual/user the title of "King of Ether" upon payment of a specified amount of ether requested by the current reigning king. As depicted in figure - 4, the king receives profit (compensation) by exploiting the discrepancy between the price he paid to his predecessor and the price others pay to become his successor. Upon a user's claim to the throne, the contract transfers the compensation to the abdicated king and designates the user as the successor king. The KoET contract fails to verify the outcome of the compensation transaction prior to appointing the new King. If the transaction fails to complete successfully, the King will not only lose his throne but also the profit. Consequently, this would ultimately result in the termination of the Kingdom of Eternal Thrones. This issue typically arises due to the lack of knowledge of the future monarch regarding the appropriate allocation of the gas quantity for the transaction. Prior to executing the logic, the contract is designed to verify the success of the transaction [1].

Source code -2

```
1. contract King_Of_The_Ether_Throne {
2.   struct Monarch {
3.   // address of the king .
4.   address eth_Addr ;  string name ;
5.   // how much he pays to previous king
6.   uint claim_Price ;  uint coronation_Timestamp ;
7.    }
8.    Monarch public current_Monarch ;
9.    // claim the throne
10.  function claim_Throne ( string name ) {
11.  if ( current_Monarch . eth_Addr != wizardAddress )
12. current_Monarch . eth_Addr . send ( compensation );
13. // assign the new king
14. current_Monarch = Monarch (
15. msg. sender , name ,
16. valuePaid , block . timestamp );
17.  }}
```

The above pseudo code is for a very famous game of king of ether throne. The struct in line 2 stores address of the present king, name, the price he paid to become king and the timestamp when he became king. The function claim_Throne on line 12 is called when any users tries to get the throne by sending compensation to current king. But in line 15, there's no check on if the call is successful or returned with an exception.

D. Parity Wallet hack

In this analysis, we examine a malicious attack that occurred in November 2017, resulting in the freezing of an account containing Ether equivalent to 162 million USD [4]. The attacker gained control of a library contract, terminated it, and obtained functionalities of 500 multi-signature wallets, leading to the irreversible freezing of the funds. Remarkably, the actions taken by the attacker were entirely legal. By initializing a previously uninitialized wallet, the attacker acquired all the privileged functions of the contract, including the kill() function. Typically, the kill() function invokes suicide(), which transfers the remaining funds to the owner and destroys the contract, thereby clearing its storage and code. However, in this instance, the attacker replaced it with self-destruct(), causing the library to return false. Consequently, all multi-signature wallet contracts were set to zero. Although the contract still held the funds, the library code was nullified. This action locked all the signatures, and the majority of the functionality dependent on the library returned zero for any function call [4].Proposed Model for the development of secure smart contract. We used the Rock-Paper-Scissor game concepts to demonstrate the proposed smart contract development to defense against attacks mention in section 6.0 .

1. Two players input their choice (Rock, Paper, Scissors)
2. As the smart contract is executed on an open network, a player can view the other's answer and hence can change his/her answer accordingly.
3. To counter this, the input should not be plaintext, but rather a hash of the choice and the nodes public address.
4. After both, the players have given their input and they are locked, then they are asked again for their choices (This time plaintext).
5. The hashes are recalculated with the node's address.
6. If the hashes are similar it's a fair game, otherwise, it's not. As shown in the Figure – 2 state diagram and source code - 3 for a secure rock paper scissors distributed game. The init function initializes the check_winner 3*3 matrix, which given the player's choices, returns the winning player. A player sends the currency and commits her choice. The function addplayer checks if the number of players till now is less than two and if the value is at least 1000. The commitment argument is stored in the contract. Any of the two players can request open by calling the open function. The open function checks if the player hasn't already committed and verifies the hash and starts the timer. The two players must be given at least 10 blocks between them to open. Finally, the check function verifies that both players have committed. The winner is decided by the check_winner matrix. If either of the players hasn't revealed her choice, the reward is automatically sent to the other.

Source code - 3

```
# Function to register a player's hashed commitment
def registerCommitment(commitmentHash):
    if self.activePlayers < 2 and msg.deposit >= 1000:
        self.pot += msg.deposit
        self.contestants[self.activePlayers].wallet = msg.origin
        self.contestants[self.activePlayers].pledge =
            commitmentHash
        self.activePlayers += 1

# Function to validate and reveal a player's choice
def revealSelection(selection, secretKey):
    # Ensure the hash matches the previously committed one
        and the choice hasn't been revealed yet
    if sha3([msg.origin, selection, secretKey], items=3) ==
        self.contestants[playerIndex].pledge and not
        self.contestants[playerIndex].revealed:
        # Store the player's choice openly if the commitment
        checks out
        self.contestants[playerIndex].decision = selection

# Function to determine the game outcome
def determineOutcome():
    # Ensure both players have disclosed their choices
    if self.contestants[0].revealed and
        self.contestants[1].revealed:
        firstChoice = self.contestants[0].decision
        secondChoice = self.contestants[1].decision

        # Determine the winner based on the choices
        if self.decisionMatrix[firstChoice][secondChoice] == 0:
            transferFunds(0, self.contestants[0].wallet, self.pot)
            return 0
        elif self.decisionMatrix[firstChoice][secondChoice] ==
            1:
            transferFunds(0, self.contestants[1].wallet, self.pot)
            return 1
        else:
            # Split the pot in case of a draw
        transferFunds(0, self.contestants[0].wallet, self.pot / 2)
        transferFunds(0, self.contestants[1].wallet, self.pot / 2)
```

return 2

VII. Conclusion

The blockchain is a decentralised ledger that records all transactions or digital events across a peer-to-peer network. The potential uses of blockchain technology encompass various domains such as crypto-currency, voting systems, healthcare management, fund transfers, and more. A smart contract enables secure transactions between unrelated and unidentified parties without the requirement of a central authority, functioning as an autonomous and self-executing agreement. These smart contracts are susceptible to attacks due to their execution in an open network. This research paper explores various types of blockchain, their significance, the fundamental principles of smart contracts, and their relevance, as well as the pivotal role of smart contracts in the field of cryptocurrency. Our literature review reveals that while researchers have conducted studies in the field of attacks, only a limited number of them have specifically addressed resource-exhaustion attacks, transaction-ordering dependence attacks, exception disorder attacks, and parity wallet hacks. The research paper thoroughly examines the attacks and presents a methodology for defending against them. The proposed defence strategy is implemented using Go-ethereum, Homebrew, and Solidity.

References

1. Luu, L., Chu, D-H., Olickel, H., Saxena, P., Hobor, A. ,Making smart contracts smarter, In Proceedings of ACM SIGSAC Conference on Computer and Communications Security, 2016
2. Nick Szabo, Smart Contracts: Building Blocks for Digital Markets, https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html. [Accessed: July 27, 2020]
3. Lin and Liao, 2017 I.-C. Lin, T.-C. Liao A survey of blockchain security issues and challenges IJ Network Security, 19 (5) (2017), pp. 653-659.
4. G. Destefanis, M. Marchesi, M. Ortu, R. Tonelli, A. Bracciali, and R. Hierons, ‘‘Smart contracts vulnerabilities: A call for blockchain software engineering?’’ in Proc. Int. Workshop Blockchain Oriented Softw. Eng. (IWBOSE), Mar. 2018, pp. 19–25.
5. Zykov S. Managing Software Crisis: A Smart Way to Enterprise Agility. Springer Series in Smart Innovation, Systems and Technologies. Switzerland: Springer International Publishing, 2018, 158pp.
6. Ingole, M. K. R., & Yamde, M. S. (2018). Blockchain Technology in Cloud Computing: A Systematic Review.
7. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. http: //bitcoin.org/bitcoin.pdf, 2008. [Accessed: July 27, 2020]
8. 6 Essential Blockchain TechnologyConcepts You Need To Know https://tradeix.com/essential-blockchain-technology-concepts/. [Accessed: July 27, 2020]
9. Z Zheng, S Xie, H Dai, X Chen, H Wang, in 2017 IEEE International Congress on Big Data (BigData Congress). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends (2017), pp. 557–564 https://doi.org/10.1109/BigDataCongress.2017.85
10. Z. Zheng, S. Xie, H. Dai, H. Wang, "An overview of blockchain technology: Architecture consensus and fu-ture trends", Proc. IEEE Int. Congr. Big Data Big Data Congr., pp. 557-564, Jun. 2017.
11. S. Singh and N. Singh, "Blockchain: Future of financial and cyber security" , in Proc. 2nd Int. Conf. Contempo-rary Comput. Inf. (IC3I), Noida, India, Dec. 2016, pp. 463467
12. Wüst K., Gervais A. Do you need a blockchain? IACR Cryptology EPrint Archive, 2017 (2017), p. 375
13. G. Wood. Ethereum: A secure decentralized transaction ledger. http: //gavwood.com/paper.pdf. [Accessed: July 27, 2020]
14. Singh, R. M. Parizi, Q. Zhang, K.-K.-R. Choo, and A. Dehghantanha, ‘‘Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities,’’ Comput. Secur., vol. 88, Jan. 2020, Art. no. 101654.
15. Zhao et al., 2016 J.L. Zhao, S. Fan, J. Yan Overview of business innovations and research opportunities in blockchain and introduction to the special issue Financial Innovation, 2 (1) (2016), p. 28.
16. Kevin Delmolino, Mitchell Arnett, Ahmed E Kosba, Andrew Miller, and Elaine Shi. 2015. Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab. IACR Cryptology ePrint Archive 2015 (2015), 460.
17. Szabo, N. (1997) Formalizing and Securing Relationships on Public Networks. First Monday, 2, No. 9.
18. Beck, R., Stenum Czepluch, J., Lollike, N. and Malone, S. (2016) Blockchain the Gateway to Trust-Free Cryptographic Transactions. 24th European Conference on Information Systems, Istanbul, Turkey, 1-14.
19. M. Vukolić, “Rethinking permissioned blockchains," in Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, BCC '17, pp. 3-7, ACM, 2017.
20. Tschorsch, F. and Scheuermann, B. (2016) Bitcoin and Beyond: A Technical Survey on Decentralized Digital

Currencies. IEEE Communication Survey Tutorial, 18, 2084-2123. https://doi.org/10.1109/COMST.2016.2535718.

21. Karamitsos, M. Papadaki, N. Baker Al Barghuthi, "Design of the Blockchain Smart Contract: A Use Case for Real Estate", Journal of Information Security, No. 9, pp. 177-190, 2018.
22. Lim, I.K., Kim, Y.H., Lee, J.G., Lee, J.P., Nam-Gung, H. and Lee, J.K. (2014) The Analysis and Countermeasures on Security Breach of Bitcoin. In: International Conference on Computational Science and Its Applications, Springer International Publishing, Berlin, 720-732. https://doi.org/10.1007/978-3-319-09147-1_52.
23. Zyskind, G., Nathan, O. and Pentland, A. (2015) Decentralizing Privacy: Using Blockchain to Protect Personal Data. IEEE Security and Privacy Workshops, San Jose, 180-184.
24. Luu L, Teutsch J, Kulkarni R, Saxena P. Demystifying Incentives in the Consensus Computer. In: Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security. CCS'15. New York, NY, USA: ACM; 2015. p. 706–719. Available from: http://doi.acm.org/10.1145/2810103.2813659.
25. Beikverdi A, Song J. Trend of centralization in Bitcoin's distributed network. In: Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on; 2015. p. 1–6.
26. Vasek M, Thornton M, Moore T. Empirical Analysis of Denial-of-Service Attacks in the Bitcoin Ecosystem. In: Bhme R, Brenner M, Moore T, Smith M, editors. Financial Cryptography and Data Security. vol. 8438 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2014. p. 57–71. Available from: http://dx.doi.org/10.1007/978-3-662-44774-1_5.
27. Eyal I, Sirer E. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. In: Christin N, Safavi-Naini R, editors. Financial Cryptography and Data Security. vol. 8437 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2014. p. 436–454. Available from: http://dx.doi.org/10.1007/978-3-662-45472-5_28.
28. Armknecht F, Karame G, Mandal A, Youssef F, Zenner E. Ripple: Overview and Outlook. In: Conti M, Schunter M, Askoxylakis I, editors. Trust and Trustworthy Computing. vol. 9229 of Lecture Notes in Computer Science. Springer International Publishing; 2015. p. 163–180. Available from: http://dx.doi.org/10.1007/978-3-319-22846-4_10.
29. Decker C, Wattenhofer R. Bitcoin Transaction Malleability and MtGox. In: Kutyowski M, Vaidya J, editors. Computer Security—ESORICS 2014. vol. 8713 of Lecture Notes in Computer Science. Springer International Publishing; 2014. p. 313–326. Available from: http://dx.doi.org/10.1007/978-3-319-11212-1_18
30. Andrychowicz M, Dziembowski S, Malinowski D, Mazurek . On the Malleability of Bitcoin Transactions. In: Brenner M, Christin N, Johnson B, Rohloff K, editors. Financial Cryptography and Data Security. vol. 8976 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2015. p. 1–18. Available from: http://dx.doi.org/10.1007/978-3-662-48051-9_1..
31. Bos JW, Halderman JA, Heninger N, Moore J, Naehrig M, Wustrow E. Elliptic Curve Cryptography in Practice. In: Financial Cryptography and Data Security—18th International Conference, FC 2014, Christ Church, Barbados, March 3–7, 2014, Revised Selected Papers; 2014. p. 157–175. Available from: http://dx.doi.org/10.1007/978-3-662-45472-5_11.