Empirical Analysis of Transaction Malleability within Blockchain-based e-Voting

Kashif Mehboob Khan[a], Junaid Arshad[b], Muhammad Mubashir Khan[c]

[a]*Department of Software Engineering, NED University of Engineering & Technology Karachi, Pakistan*
[b]*School of Computing and Digital Technology, Birmingham City University, Birmingham, UK*
[c]*Department of Computer Science & IT, NED University of Engineering & Technology Karachi, Pakistan*

Abstract

Blockchain is a disruptive technology that has been used to address a wide range of challenges in diverse domains including voting, logistics, healthcare and finance. Transaction malleability is one of the critical threats for blockchain, which can facilitate double-spending attacks by tampering with the state of a blockchain. This paper investigates the potential of transaction malleability attack within a blockchain-based application (e-voting) with the aim to identify settings which can lead to a successful transaction malleability attack. Therein, we aim to highlight conditions which can cause transaction malleability attack so as to help develop appropriate protection mechanisms. In particular, a successful execution of transaction malleability attack is presented which was conducted on a blockchain testbed hosting an e-voting application. The experiments identified significance of parameters such as network delay and block generation rate to successfully execute transaction malleability attack and have highlighted directions for future research.

Keywords: Blockchain, Transaction Malleability, Blockchain Security, Security Evaluation, e-Voting

1. Introduction

Blockchain is one of the disruptive technologies of the current era, which enables trustworthy transaction processing through participation of distributed nodes. The core data structure used within a blockchain resembles a linked list shared among all nodes of a network where each node keeps a local copy of all the blocks starting from a *genesis* block [1]. Being the most notable blockchain-based application, the success of Bitcoin has introduced an innovative model of application development across diverse domains such as healthcare, e-voting [2] and e-document management [3]. Such applications leverage benefits of the blockchain technology due to its cryptographic validation structure among transactions as well as public availability of distributed ledger of transaction-records in a peer-to-peer network. Therefore, creating a chain of blocks connected by cryptographic constructs makes it very difficult to tamper records within a blockchain, as it will require rework from the genesis block to the latest transaction in blocks [4].

Public voting is a fundamental component of modern democracies and has received significant attention to improve its processes as well as use of technology (*electronic or e-voting*)to enhance its accessibility and acceptance for large-scale participation. A major concern for e-voting systems is the trustworthiness of the voting processes as well as the security of the e-voting application and its resilience against emerging threats. This raises several new technical concerns, such as the security of governance processes within an e-voting system, the security of the e-voting software against application-level threats, the protection mechanisms implemented to achieve secure, and tamper-proof auditing. Chaum et al. [5] summarized such concerns by prescribing fundamental requirements to achieve end-to-end (E2E) verifiability for an e-voting system aiming to facilitate rigorous evaluation of such systems. Consequently, various efforts have been made by scientific community to develop e-voting systems which can satisfy the requirements to achieve E2E verifiable e-voting.

Blockchain technology can facilitate developing E2E verifiable e-voting system by increasing the difficulty of re-using the token (vote). This is because in order to be part of a blockchain, all the nodes are required to be initialized from the same genesis block to add transactions into the blockchain ledger shared among them. Therefore, as every proof of work for validating a vote transaction will be computed based on all the transactions from the genesis block to the latest block, it can increase the cost of double utilization of a token (vote) exponentially. Moreover, since blockchain technology is decentralized, each node will be running their local copy of the blockchain which may later be used for consensus as elaborated in [2]. A number of efforts have been made to explore the use of blockchain technology to achieve trustworthy E2E e-voting with [6, 7, 8] representing latest advancements in this respect.

However, several vulnerabilities have been highlighted in the blockchain fabric, leading to a number of efforts to review and identify security threats for blockchain. For instance, Li et al. [9] carried out a thorough study on the

major attacks on blockchain, highlighting loopholes in the Proof of Work (PoW) based consensus system with respect to $> 50\%$ attack. They illustrated how the control over mining power by a single entity or group of entities may cause significant damage to a major feature of blockchain i.e. decentralization. Similarly, other attack vectors such as encryption scheme used in the transactions, methods for verifying transactions and the design of the transaction have been identified as sources of threats for blockchain-based applications. Furthermore, Iuon-Chang [10] analyzed various security challenges in blockchain which can be used to carry out attacks, specifically highlighting the risk of $> 50\%$ attack, forking problems and scalability of blockchain with significant impact on blockchain security.

The focus of our research is to investigate challenges in the adoption of blockchain to achieve decentralized trustworthy applications with specific emphasis on challenges impacting security of a blockchain. In this paper, we are focused at the challenge of *transaction malleability* for blockchain-based applications which can lead to an inconsistent blockchain state potentially resulting in further attacks, such as double-spending. In particular, we aim to identify settings which may lead to a successful transaction malleability attack, thereby highlighting conditions causing such attack to facilitate the development of protection mechanisms for them. In order to conduct rigorous empirical analysis of transaction, we have established a blockchain testbed using Multichain [11] which makes use of multiple participant nodes to simulate a typical real-world blockchain system. Furthermore, we chose e-voting as an example application to facilitate our experimentation, making use of the blockchain-based e-voting system proposed in [2]. Specifically, this paper makes the following contributions:

1. An in-depth classification of security threats for blockchain-based systems is presented. This classification takes into account different sources of attack as well as their target layer i.e. core, network or mining.
2. In order to analyse the fundamental mechanism and the underlying causes of a transaction malleability attack, an attack model has been executed on a real blockchain using a public voting scenario. It facilitates assessing impact of transaction malleability on a real-world blockchain and in the context of e-voting, which is one of the prominent applications of blockchain.
3. A testbed simulating real-world public e-voting scenario has been implemented which enables rigorous evaluation of the feasibility and impact of transaction malleability attack for a blockchain-based application.
4. The trade-off between security and performance (scalability) for a blockchain-based application (public e-voting) is highlighted through rigorous experimentation on a real-world blockchain testbed.

Rest of the paper is organized as follows. Section 2 presents a summary of the leading threats for blockchain-based applications focusing specifically on the threats causing transaction malleability. Section 3 presents a critical review of the existing efforts made to investigate security threats within blockchain in general and transaction malleability in particular. It also includes a summary of recent efforts to use blockchain technology to achieve E2E verifiable e-voting applications. Section 4 presents background knowledge in two important aspects of the paper; i) Multichain blockchain platform which has been used for the implementation and experimentation in this paper, and ii) a brief description of our e-voting model summarizing our previous work [2]. In section 5, we present the specific attack model designed to implement our proposed public voting model and conduct transaction malleability attack to aid our investigation. Section 6 presents the details of implementation of our proposed e-voting model with blockchain, including step-by-step function and interaction between different entities. Section 7 includes details of the experimentation performed as part of our investigation for transaction malleability attack along with analysis of the behaviour observed. Section 8 presents a brief summary of our ongoing research into developing a protection mechanism for transaction malleability attack followed by 9 which concludes the paper.

2. Security Threats in Blockchain

Due to the increasing use of blockchain for diverse application domains, the volume of attack attempts on blockchain has also increased significantly, leading to a number of efforts to identify, and mitigate against specific attacks. For instance, Li et al. [9], Lin et al. [10], and Lu et al. [12] are some of the efforts focused at investigating the feasibility and significance of $>50\%$ attack for a blockchain-based system. They have focused at weaknesses within PoW consensus algorithm, inconsistencies in data due to forking, and scalability management as some of the causes of such attacks. However, to the best of our knowledge, the impact of transaction malleability attack for blockchain-based systems in general and for blockchain-based e-voting in particular has not been investigated at great depth. The impact of such attack on an e-voting system is envisioned to be significant especially due to potential disruption to voting process in the event of a successful transaction malleability attack.

A classification of security threats for blockchain-based systems is presented in Fig 1 which categorizes variety of commonly known attacks that can affect blockchain networks. The three classes include

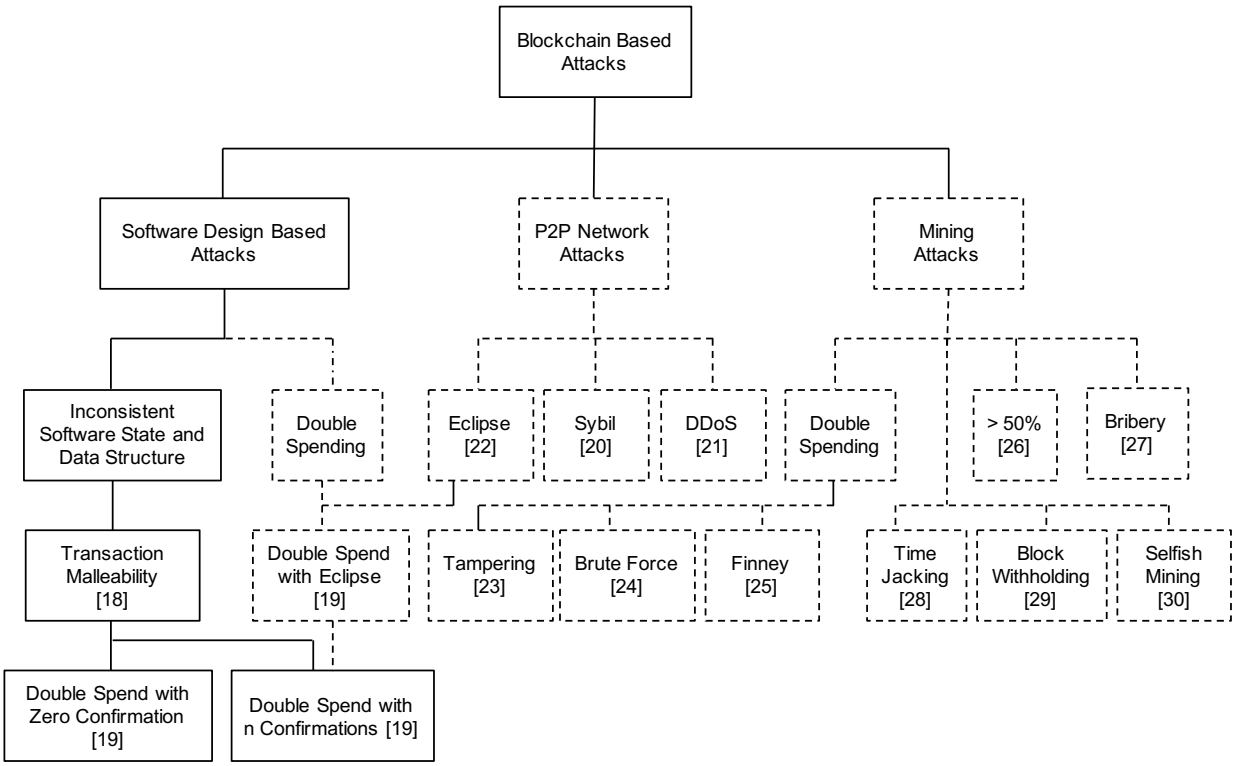


Figure 1: Security threats for blockchain-based systems.

Software Design-based Attacks, *P2P Network Attacks*, and *Mining Attacks* and have been discussed below.

2.1. Software Design-based Attacks:

Software design-based attacks refer to manipulation of the blockchain state by exploiting software vulnerabilities within the fundamental design of blockchain such as its data structure, and manipulating various transaction fields for malicious activities. In a typical transaction malleability attack, the transaction ID is changed before it gets mined in the blockchain network. Typically, the aim is to make use of this confirmed malleable transaction in repeating the original transaction which could not be mined first [13, 14]. One of the most significant challenges is the double spending attack [15, 16, 17, 18], which enables an attacker to create an illusion of presenting and reusing an asset (or any token). Bitcoin is the first widely accepted cryptocurrency which proposed a scheme to prevent double spending [15, 16, 17, 18] by utilizing peer-to-peer distributed network technology for maintaining the order of the transactions.

2.2. P2P Network Attacks:

Through a P2P Network attack, an attacker can take control of the blockchain network and perform manipulation with its network scope potentially disrupting its property of decentralization. For instance, Eclipse attack [19] works by taking control of the communication of victim's computer thereby spoofing the state of the network. The attacker nodes in such case may write their addresses to the neighbour's list of benign nodes. In a successful eclipse attack, an attacker takes control of the nodes and develops a spoofed image of blockchain for those nodes by controlling their network traffic's incoming and outgoing communications. The mining power is therefore wasted in computing a spoofed blockchain.

In a typical blockchain, the consistency of blockchain is maintained by keeping an updated local copy of all the nodes to a state acknowledged by majority of the miners. This process is vulnerable to Sybil attacks [20] where an attacker creates several virtual nodes in the network to sabotage the consensus process by agreeing on a malicious transaction.

In DDoS attack [21], after gaining control of several nodes of a network, an attacker starts redirecting these control nodes to a victim resource they wish to attack. All of this is done transparent to the nodes which are under attack. In the context of blockchain, DDoS causes the network potential to be drained resulting in the denial of service to the honest miners.

2.3. Mining Attacks

Mining attacks involve miners to make use of the mining process for their own interest by accepting or rejecting transactions. These miners usually work in collaboration with each other to achieve their common

objective and disturb the benign objective of the network i.e. to add transactions.

For instance, tampering attack [22] causes some nodes to receive late updates about transaction blocks to be added to their local copy of data. This situation may result in serious consequences including denial of service and double spending attacks [16]. Brute force attack [23] uses trial and error approach to decrypt or recover any hidden or secret information to fulfil malicious objectives. In the context of blockchain, the attacker keeps on mining on blockchain forks to cause double spending by gradually increasing the size of the chain. The aim is to deprive the merchant of the assets they own. Finney attack [24] is a type of mining attack where selfish miners make use of pre-mined blocks by broadcasting them to the network to negate the honest transaction of a miner. $> 50\%$ attack [25] is aimed to disrupt the consensus rule of the blockchain network which creates a scenario where the attacker takes control of majority of the mining power. In bribery attack [26], the attacker attempts to bribe other miners for mining their own interest aiming to conduct double spending [16]. Time jacking attack [27] disturbs the miner's clock which can ultimately result in disturbing the difficulty level of mining process. In block withholding attack [28] [29], the profit of the pool is reduced by not broadcasting the blocks. Selfish mining [30] aims to form an unfair strategy to win the rewards for mining by selfish miner. In this case, a miner or its pool stops issuing the solution to the network and continues to mine to later on add their chain as a longer chain than the existing one, so that this may be acknowledged by the network, resulting in wasting the working of the honest miners.

3. Related Works

Double spending attack is the possibility to spend a transaction more than once and can be caused by number of ways such as by; successfully committing malleable transactions, controlling more than 50% of mining power of network by an attacker or, attaining enough computational power to be able to create a blockchain longer than the benign blockchain accepted by the majority of the blockchain nodes. Among these, transaction malleability has received increased attention primarily due to the Mt. Gox attack in 2014. Since then, a number of efforts have been made by research community to analyze the feasibility of such attack in real-world blockchain scenario as well as approaches to mitigate the double spending problem in blockchain. In this section, we first summarise significant efforts with respect to analysing and mitigating transaction malleability in blockchain. Further, we present an overview of the significant efforts to achieve E2E verifiable e-voting systems using blockchain.

3.1. Transaction malleability in blockchain

In order to evaluate the impact of transaction malleability in blockchain, a potential avenue is to artificially inject multiple malleable transactions immediately after an honest transaction by changing the non-functional fields i.e. parameters that do not change the semantics of the transaction, such as *sender's address, receiver's address, and amount.* Although these fields are cryptographically signed by the sender, a modification in these can result in changing the hash of the transaction ID (TxID). Such efforts exemplify malicious attempts by attackers to identify and exploit vulnerabilities to achieve transaction malleability and, therefore, require efforts to develop appropriate protection mechanisms. For instance, the vulnerability in consensus regarding OpenSSL was fixed by BIP66 and has been active from block 363,724 which was added to the blockchain on July 4, 2015 [31]. Similarly, Bitcoin's use of Elliptic Curve Digital Signature Algorithm (ECDSA) to generate transaction IDs has been attacked to the effect that new hashes (transaction IDs) may be formed against substantially similar transaction. If any of the malleable transactions is mined before the original transaction, the miners will add this transaction to the block as the valid transaction because the critical fields in the transaction were unchanged. However, if the sender of the transaction looks for the confirmation of the transaction by its transaction ID in the transaction database (publicly shared blocks), they will not find it as the original transaction would be rejected by the miner as a double-spend [32].

Within this context, the solution proposed by Andrychowicz and Dziembowski [1] for handling transaction malleability requires elimination of script of the transaction which may cause malleability. The script is removed before the computation of hash and is therefore considered a significant performance overhead. Decker and Wattenhofer [14] investigated the suspected transaction malleability attack of Mt. Gox to identify how it caused double-spend in the Bitcoin. Although they carried out an analytical study, the authors did not provide any specific solution to the problem of transaction malleability. Rajput et al. [33] presented a solution to address the transaction malleability challenge in Bitcoin through alterations in the process of computing a final transaction hash to identify a transaction. Specifically, they proposed calculating hash value without using *scriptSig* field of a transaction as this field gives an attacker the liberty to add or replace keywords which do not change the semantics of the transaction. Although the authors proposed a scheme to address signature malleability for Bitcoin, the solution has limitations concerning its integration with blockchain-based applications in wider domains.

Segregated Witness (also known as SegWit) [34] was initially proposed as a solution to address scalability challenge in Bitcoin; however, it is also considered a

defence against transaction malleability attack caused by signature malleability. This is because the organization of data for signature of transaction is separated, and not a part of the transaction called *witness*. Due to this, it is also considered as a useful barrier to successful execution of transaction malleability attack. One of the major problems of using SegWit [34] is the compatibility i.e. the transactions following this scheme remain separated from the conventional transactions of blockchain and has split the Bitcoin community into two groups; Bitcoin and Bitcoin Cash (BCH). As BCH community does not use SegWit, transaction malleability is still a challenge in Bitcoin cash.

3.2. Blockchain-based e-voting

From the perspective of e-voting, a number of attempts have been witnessed to disrupt electoral processes. Consequently, researchers have investigated use of technology to strengthen such democratic processes leading to development of major projects, such as My Vote [35] and BitCongress [36]. However, the protocols used are ambiguous in keeping their essential properties, and their scientific contribution is limited. Another monetary-based system for e-voting was introduced by Zhao and Chan [37] where reward is given by enforcing smart contracts over the distributed network but it affects the actual freedom of ballot. Recently, there have been several approaches in blockchain-based online voting which provide secrecy of the ballot including [38] and [39] which work on the secrecy of ballot.

More recent efforts using blockchain to aid e-voting have focused on leveraging blockchain properties to achieve verifiable voting such as [40]. Furthermore, confidentiality of the individual vote is a fundamental component of a voting system however default settings within blockchain do not fulfil this requirement. Consequently, recent efforts such as [41] have focused at achieving privacy-aware voting overlaying blockchain technology to fulfil this requirement. With respect to double utilization in e-voting, existing approaches rely on the assumptions that majority of computing resources are controlled by honest miners. Furthermore, our earlier work [42] presented initial efforts to investigate transaction malleability within a blockchain-based e-voting application within a Python-based simulated blockchain environment. This highlighted the feasibility of such attack within a simulated blockchain environment providing motivation to explore it further in a real-world blockchain environment.

We believe that the processing sub-domain of electronic voting is an area where further scientific contributions are required to specifically target the potential of double utilization of tokens in a blockchain-based decentralized network. In this context, transaction malleability attack can serve as a platform for further malicious activities such as blockchain-forking as well as misleading the voter that their vote has not been caste. Therefore, our focus in this paper is to investigate execution of transaction malleability attack to highlight the need for further research concerning protection mechanisms to mitigate against this threat.

4. Background: Blockchain Platform and e-Voting System

In order to assess the feasibility of transaction malleability attack, we have used Multichain as the blockchain platform and the e-voting system developed as part of our existing research presented in [2]. A brief account of these is presented below to provide context for the rest of the paper.

4.1. Multichain

Multichain is an open-source platform that helps users to develop a wide range of blockchain-based applications. Multichain provides a collection of simple commands to configure and set up blockchain. One of the major objectives of Multichain platform is to help develop the controlled and private blockchain network. Being a private blockchain by default, Multichain may be customized to reduce the compute-intensive work of mining process with a variety of options. This model of blockchain transacts only against the approved accounts of this chain. If the Multichain blockchain is set as private, challenges involved in mining can be easily addressed using various configuration options in this platform. To recognize the agreements in Multichain, public-key cryptography is used where each user is assigned a public key in order to achieve the identity information and safety of the user. Users create their own private key which is not exposed to other members. Each private key has a mathematically related public address which contains information to receive funds as part of the transactions.

Leveraging fundamental capabilities of public-key cryptography, Multichain achieves authenticity (through digital signatures), confidentiality of communication (through use of SSL/TLS), and data integrity (through use of cryptographic hashes). We have used these capabilities of Multichain to achieve secure, verifiable communication within our e-voting model.

4.2. Blockchain-based e-voting

As part of our previous research [2], we implemented a blockchain-based e-voting system which enables secure verifiable public voting in a tamper-proof manner. Fig 2 presents a graphical representation of this model.

Overall, the electoral process involves execution of certain activities in a pre-defined chronological order. This set of activities may be categorized into offline and online processes. The offline tasks include the creation of voters and candidates' addresses, setting up their roles and respective authorization in accordance with their

polling stations and other associated parameters. The offline process also manages the data of voters from blockchain and creates station wise lists to be later transmitted to their respective electronic voting machines in accordance with the location of polling stations. The back-end data processing supports a conventional public voting model through a permissioned blockchain to regulate entities such as voters, candidates, and miners as per the procedure of general public voting model while executing real time online e-voting activities.

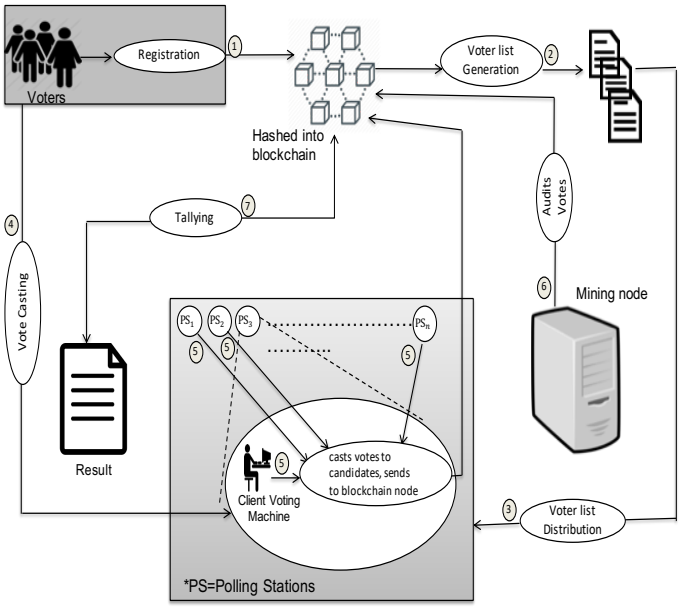


Figure 2: A blockchain-based e-voting system.

5. Attack Model

A typical transaction malleability attack aims to exploit vulnerabilities within blockchain fabric and therefore its impact across different application domains is consistent. Consequently, although cryptocurrency applications overshadow blockchain-based applications, transaction malleability can have adverse impact on wider application domains including e-voting, supply chain management, healthcare, and document verification. In this section, we present the theoretical and practical considerations for this attack.

5.1. Theoretical considerations

Within a typical transaction malleability attack scenario, we assume two transactions i.e. an honest transaction *Th* and a semantically equivalent malleable or malicious transaction *Tm*. These transactions are submitted to the blockchain network at times t_h and t_m respectively where $t_h < t_m$. The goal of the malicious attacker in this case is to tamper with normal working of the blockchain network in a way so that the Tm is mined ahead of Th. Leveraging constructs defined by Narayanan et al. [43], we can use Poisson distribution to determine the possible efforts or level of progress, the attacker has made. In this case, the acknowledgement for a vote is not sent unless the transaction becomes a part of the block and even z more blocks are added to it with the condition that honest blocks are being added at average expected time per block [44]. The expected value for the attacker's progress in this case will be:

$$\lambda = z(p/q) \tag{1}$$

Where p shows the probability of finding the next block by an honest miner while q denotes the probability and mining strength of attacker to find the next block. This expected value λ can be calculated using Poisson experiment [44] by taking into consideration the total count of successful attacks in a sequence of time-bounded activity. Since the expected value λ is being computed on the basis of Poisson's experiment, the occurrence of successful events during an interval are completely independent of each other. Also, the probability of a single successful event in an interval varies according to the length of the interval. The model also applies uniform probability of success during the execution of experiment i.e. miners cannot change their resources and computing power [17].

Since λ is the rate of successful events occurring during each interval and we are interested in the case where an attacker is capable to add their blocks to the blockchain, a success in this case will reflect the number of blocks an attacker can discover. Similarly, an interval refers to the amount of time, the election official waits for z blocks to be added by the honest miner. Therefore λ will be measured in number of blocks discovered per unit interval.

Applying the implications of Poisson's experiment from double spend attack in Bitcoin system (where the difficulty level is adjusted continuously to maintain an average time of 10 minutes per block) to double utilization of vote in electronic voting, we can easily infer that an honest node will add p blocks in every t minutes. Therefore z blocks will be added at an interval of

$$interval = z(t/p)min \tag{2}$$

Since it is a competition between an honest miner and an attacker to get their transaction mined first, the attacker produces 'q blocks for every t minutes. Mathematically, the rate at which blocks are addded by an attacker R can be represented as:

$$R = ((qblocks)/(tminute)) \tag{3}$$

Therefore, the average number of expected successes λ during interval will be obtained by multiplying eq. 1 and 2 as below:

$$\lambda = (z(t/p) * (q/t)) = (zq/p) \tag{4}$$

Now, if we are aiming at a particular case to find out X successes in a trial where $X = k$ successes and k is a non-

negative real number; $k \geq 0$ may be obtained as;

$$p(X = k; \lambda) = \frac{(\lambda^k e^{-\lambda})}{k!} \tag{5}$$

Consequently, the probability of the attacker in accordance with our e-voting attack model may be computed with the above equation when an attacker is producing k block in an interval for which z blocks are produced by honest nodes.

From above, eq. 4 can be used to determine the overall behaviour of the system over a particular time period under observation to analyze the number of cases for successes and failures of attacks. Further, eq. 5 can facilitate to focus on a specific favourable outcome against a certain number of attacking attempts to the blockchain. Eq. 4 and 5 can be very effective collectively for detecting the instantaneous and long-term change in the state of the blockchain as demonstrated by Narayanan et. al. in [45].

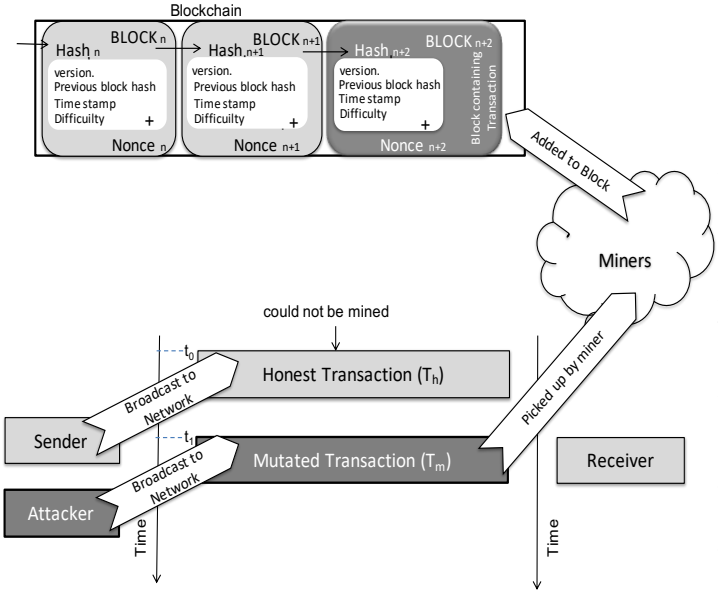


Figure 3: Simulation of attack model for transaction malleability.

5.2. Practical considerations

In order to design the attack model, we conducted a thorough review of existing studies focused at addressing the security of blockchain in general and with respect to e-voting in particular such as [46] and [47]. Specifically, Khader et al [46] addressed the issues associated with the underlying voting model such as abnormal termination of the system under a certain condition when a voter quits the voting process and about not maintaining the secrecy of the outcome of electoral process in the later stage. Similarly, other approaches which are aimed at providing secrecy of ballot in blockchain-based voting systems have been presented in [47] and [48]. These approaches rely on the assumptions that majority of the miners will be honest and would not allow double utilization of vote to be acknowledged through consensus.

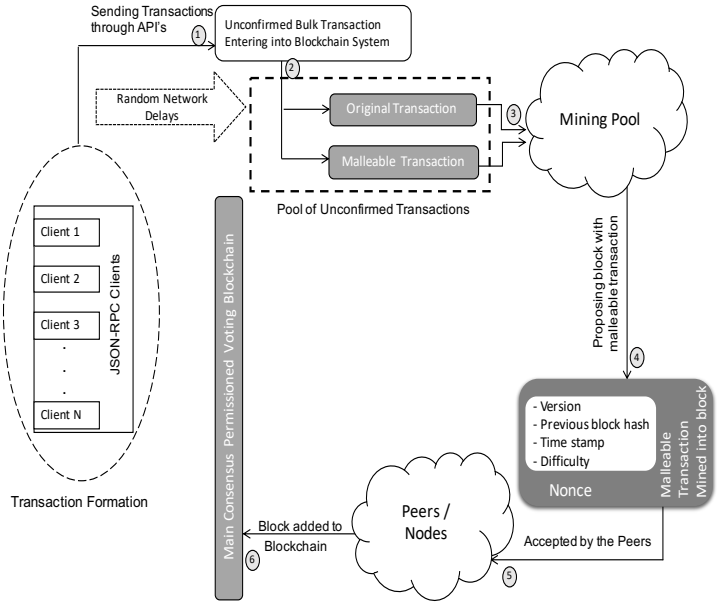


Figure 4: The process model diagram for potential transaction malleability attack.

The attack model for carrying out transaction malleability is illustrated in Fig 3 and explained as follows: Two semantically same but syntactically different transactions Th(honest transaction) and Tm (malleable transaction) are broadcasted to the blockchain network at time t_h and t_m respectively. Due to delays in the network or any other disruption caused by an attacker, malleable transaction Tm which was sent after honest transaction Th, is mined by one of the miners. The addition of blocks (event) in a blockchain is completely independent of each other and can be evaluated using Poisson distribution. There is always a chance for a mutated transaction to be mined even earlier than the original honest transaction. This scenario fulfils our requirement to carry out a successful transaction malleability attack.

Fig 3 shows the scenario of our attack model which causes a malleable transaction to get mined to a blockchain block whereas Fig 4 explains the execution details of the process employed to carry out a successful transaction malleability attack. The process can be expressed as follows:

- A network of multiple remote JSON-RPC clients is established with the capability to send large number of concurrent transactions to the blockchain network. These bulk transactions are envisaged to induce delays while arriving to the blockchain network.

- In addition to the benign transactions, the JSON-RPC clients also send malleable transactions such that there is a consistent population of malicious transactions in the pool.

- Upon entering into the pool of unconfirmed transactions, it is left to the miners to pick up either an honest transaction or its semantically similar malleable transaction. As the malleable transaction is cast from a comparatively stronger node (with greater hashing power), this helped to form a malleable transaction in a shorter amount of time to reach at a time closer to the honest transaction at the pool of unconfirmed transactions. In this way, a malleable transaction is a very healthy competitor for its equivalent honest transaction especially in the case where the malleable transaction arrives at the same time when a miner picks transaction to be mined.
- As soon as a malleable transaction is picked up by the miner and added to its proposed block, the honest version of the transaction will be automatically rejected by the miner.
- Upon acknowledging and accepting the proposed block by the nodes (seed/connected nodes) of the network, the malleable transaction is successful in becoming a part of the consensus blockchain.

6. Implementation and Experimentation Setup

In order to aid experimentation to assess feasibility of the transaction malleability attack on real-world blockchain, we have implemented a blockchain-based electronic voting system. A brief account of this system is presented in section 4 whereas further details are presented in [2] and [49]. We present discussion regarding implementation of this model with blockchain technology in this section. Fig 5 presents a graphical illustration of the implementation of proposed e-voting model with blockchain.

6.1. Blockchain parameters

The blockchain implementation consists of a *seed node* and at least one *connected node* which together form the mining pool. The rest of the machines in the network are JAVA-based remote clients using JSON-based RPC API for sending bulk transactions to the blockchain.

The **seed node** is a master node which is responsible for initiating blockchain through the genesis block. Therefore, it controls the overall behaviour of the blockchain using certain configuration parameters such as handling consensus mechanisms among nodes, setting up for permissioned/non-permissioned blockchain, rights management and other settings upon which the other peer nodes (also known as connected nodes in Multichain platform) connect and agree to be part of this blockchain with conditions as imposed by Seed node. The **connected nodes** periodically synchronize themselves with seed node to maintain the decentralization of the blockchain network. These connected nodes act as full running nodes and keep a full copy of blockchain locally. The seed and connected nodes also have the shared responsibility of forming a mining pool in the experiments as shown in Fig 5.

In view of the practical considerations of a public voting scenario, the proposed scheme has been implemented using a permissioned blockchain. Consequently, consensus algorithm such as Proof of Work (PoW) are not required for miners to add their proposed blocks to the blockchain. Furthermore, PoW incurs computational overhead with limited benefits to permissioned blockchain which led us to decide against its use. Instead, we have used two parameters i.e. *mining diversity* and *mining turnover* which are recommended by Multichain platform to control mining and develop consensus among the nodes. Blockchain setup in this paper uses a value of 0.3 for mining diversity. This implies that the minimum count of miners necessary to run the blockchain smoothly will be attained by multiplying mining diversity (which is set to 0.3 here as mentioned in Table 1) to the number of total miners available. This turns out to be 3 which means in these experiments, three miners should be available to actively participate for adding a transaction to the blockchain. Moreover, mining turnover responsible for managing scheduling of miners for their turns in a round robin fashion. Its value ranges from 0 to 1 with 1 refers to a condition in which each miner will attempt to add its proposed block to the blockchain (and may be responsible for generating forks and wastage of computational power), while a value of 0 sets up a default round robin scheduling algorithm. In these experiments, it has been set to 0.5 to maintain a stability between these two boundary value conditions.

6.2. Voting process implementation

The proposed e-voting system simulates the public voting model whereby a set of activities, such as voter and candidate registration are performed before the election/poll day. We explain the steps involved below.

1. The offline process starts by generating voter and candidate addresses at seed node which contain identification of individual voters in the form of unique hashes representing each voter and candidate respectively. The system maintains a voter list for each polling station which contains list of voters for a particular station. For our experiments, we created ninety thousand voters which are divided among thirty stations. The process of address generation was carried out using JSON-RPC-API.
2. The voter lists are requested by voting clients envisaged to be installed on designated voting machines deployed at polling stations.
3. The voter lists are distributed to machines where java-enabled voting clients are running for sending

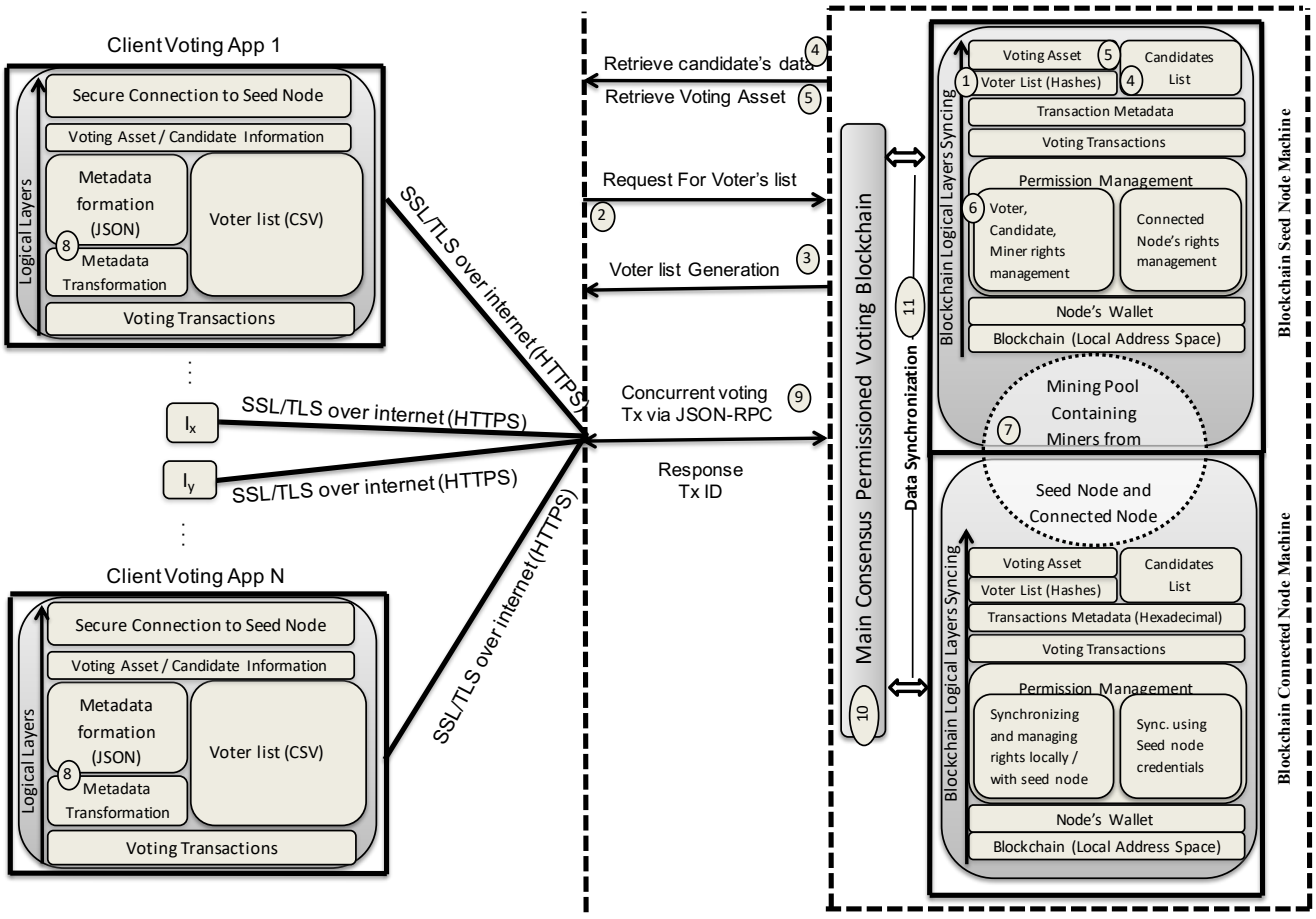
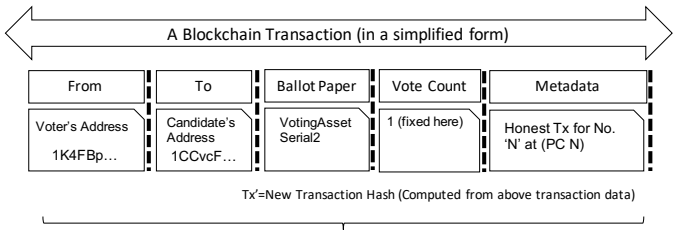


Figure 5: Test-bed to evaluate transaction malleability.

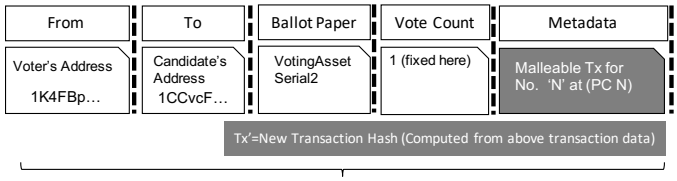
concurrent requests in bulk to the main voting blockchain powered by seed node (master node) and its connected node.

4. Generate addresses for candidates at seed node
5. Creation of voting asset at seed node
6. Grant appropriate rights for voters/candidates/vote issuing authority. This process makes up the offline activity part of voting process which can only be followed after its connected previous processes(step no 1, 4, and 5 in Figure 2)
7. Generating miners at both seed and connected nodes to be selected randomly and actively participate from the mining pool(containing only allowed miners).
8. Transaction formation including the process of creating metadata.
9. Voting clients sending requests to the voting blockchain from different concurrent client applications across different participating machines.

Additionally, Since this is a permissioned blockchain, only the addresses in the voter list will be allowed to participate in the process. Furthermore, data synchronization of voting transaction will take place at regular intervals to maintain the decentralization of the architecture. The connected node as a result of this synchronization will also update its local copy of blockchain which is located at its own address space.



(a) Honest Transaction. Structure



(b) Malleable Transaction. Structure

Figure 6: Structure for honest and malleable transactions.

Platform	Blockchain Parameters				
	Mining Diversity	No. of miners	Block generation Rate(secs)	Max. Allowable Size (MB)	Minig Turnover
Windows	0.3	10	15	8.3	0.5

Table 1: Blockchain configuration.

Node Type	Platform	Hardware Specification		
		Processor	Memory	Page File
Seed Node	Windows 10 Pro 64-bit (10.0, Build 10586)	Intel Core i3-4005u CPU @ 1.70GHz (4CPUs)	4096MB RAM	5586MB Used 1887MB available
Connected Node-I	Windows 10 Home Single Lang. 64-bit (10.0, Build 17134)	Intel Core i7-7500U CPU @ 2.70GHz (4 CPUs) 2.9GHz	8076MB RAM	14346MB used 2836 available
Connected Node-II	Windows 10 Enterprise 64 bit	Intel(R) Core(TM) i7-7500U CPU @ 3.4GHz (4 CPUs), 3.4GHz	16384MB RAM	3809MB used 14901MB available

Table 2: Hardware and software specifications.

6.3. Experimentation test-bed

In order to investigate the challenge of transaction malleability, we used the e-voting model described in section 4. There are a total of three nodes (one seed node and two connected nodes) involved in the test-bed and their significant specifications are presented in Table 2.

All of the nodes involved are live and keep a local copy of the blockchain with frequent synchronization with the main blockchain. Here, connected node-I is the weakest among the three nodes and this node will be used for sending out honest transactions followed by a simultaneous attempt of multiple malleable transactions from two relatively faster client voting machines. This will enable malleable transactions a chance to be processed and picked up by the miners of seed node before an honest transaction. This arrangement is consistent with the real-world scenarios where an attacker can establish comparatively stronger resources than an honest machine.

The experiment was divided into different trial runs. In each run, 200 honest transactions were sent from a relatively weaker node and just after the start of these honest transactions burst, an equal amount of malleable transactions against each of these honest transactions(and in the same sequence) were sent from the two relatively faster machines. For example, if there is an honest transaction *Th* there there will exist two different malleable transactions *Tm1* and *Tm2* for it from two different JAVA-based client applications on different machines.

In this way, a favourable environment is established for the malleable transaction to get mined into the block before its honest version. The malleable transaction is favoured not only in the form of the strength of the attacker machines but also the probability for the transaction malleability attack is increased with the ratio of 1:3 as one honest machine has to compete with two dishonest client machines. In this case, if even a single malleable transaction gets mined, the attack would be considered as successful. Therefore, as all the transactions are able to target the seed node directly, this provides both attacker and the honest machine equal opportunity to get their respective transaction mined. In the second set of experiment, the honest transaction will be forcefully delayed but this time from the client which is running on weaker node to observe the impact of software-level vulnerabilities (timing delay in this case).

The metadata contains the information about each machine and the nature of the transaction whether honest or malleable. Also, since it is a voting scenario and only one vote is possible for each attempt, therefore if any of the syntactically similar transaction (malleable transaction) is successful, the other transaction will automatically be discarded due to insufficient asset value (which is a vote here).

The voting transaction in the experiment consists of the following major fields: voter's address, candidate's address, voting asset serial, vote, metadata containing information about whether the sending transaction is an honest or a malleable, the transaction number(from the voter list) and the node number. This metadata is saved to the blockchain along with the transaction and can be very useful in determining which transaction (honest or malleable) was successfully mined. Fig 6 shows the structure of the transaction and its semantically equivalent malleable transaction for the experiment.

Here, it is evident that a new hash value *Tx* (transaction id) is being created to generate a malleable transaction by modifying the metadata of the original transaction. Further, metadata is not only being used to create a malleable transaction, but also to confirm the transaction that hijacks its place in the blockchain. Depending upon which transaction is mined into the block, the metadata of the respective transaction(after decoding from hexadecimal to plain text) may be used as an evidence. As presented in Fig 6, this metadata contains information about whether the transaction is honest or malleable, the order of voter's address from the

voter list and the name of the connected node that has sent this transaction. Consequently, the experiment was able to get the malleable transaction mined into the block before the honest transaction.

7. Experimentation for Transaction Malleability

This section explains the experimentation setup followed by different scenarios implemented to conduct experimentation to investigate introducing transaction malleability within blockchain.

7.1. Experimentation Setup

For our experimentation, we setup a testbed based on the technical details provided in section 6 and consisting of three nodes (one seed node and two connected nodes). The setup also included multiple JAVA-based JSON RPC remote clients which are connected through local intranet with the seed and connected nodes. In view of the sensitivity of the communication within an e-voting application, these clients use Multichain's built-in SSL libraries to achieve secure connections with other nodes in the network. Furthermore, Multichain Blockchain enables digital signature to be embedded within the metadata of a coinbase transaction. We leverage these capabilities of Multichain to ensure security and privacy of communication among nodes within our network.

The blockchain used in the experiments is private (permissioned) and only the voters from the voter list are allowed to participate. Table 1 includes the blockchain configuration which was used to carry out transaction malleability attack on the actual blockchain.

Since the honest and its associated malleable transaction was being sent almost immediately after one another, therefore this block generation rate suits the need to give the respective malleable transaction almost same chance to compete for the same block number as of its honest transaction. Similarly this configuration requires at least 30 percent of the miners to actively participate in the mining process to get their processed transaction to mine into their proposed block due to its mining diversity. This section is focused at investigating feasibility of mining a malleable transaction into a private blockchain within the context of electronic voting domain. For this purpose, a blockchain seed node was created followed by creation of the asset with the lowest value as "1", representing a single vote. This asset value will be later assigned to the address of the individual voter on the blockchain with one voting asset per voter. Figure 7 demonstrates a typical voting asset implemented within our blockchain setup and shows the asset name, its reference number, and its unit as major parameters.

Fig 8 presents important parameters for the nodes involved in the experimentation setup. Through the*pingtime* parameters of the Fig 8, it can be observed that the seed node has a lower ping time with connected


```
{
    "name" : "VotingAssetSerial2",
    "issuetxid" : "a488befbb50e53dd88bd883c79333c85dd692dbf9072950e91f11163bf2a156f",
    "assetref" : "9803-267-34980",
    "multiple" : 1,
    "units" : 1,
    "open" : false,
    "restrict" : {
        "send" : false,
        "receive" : false
    },
    "details" : {
    },
    "issueqty" : 10000000,
    "issueraw" : 10000000,
    "subscribed" : false
}
```


Figure 7: Voting asset details.

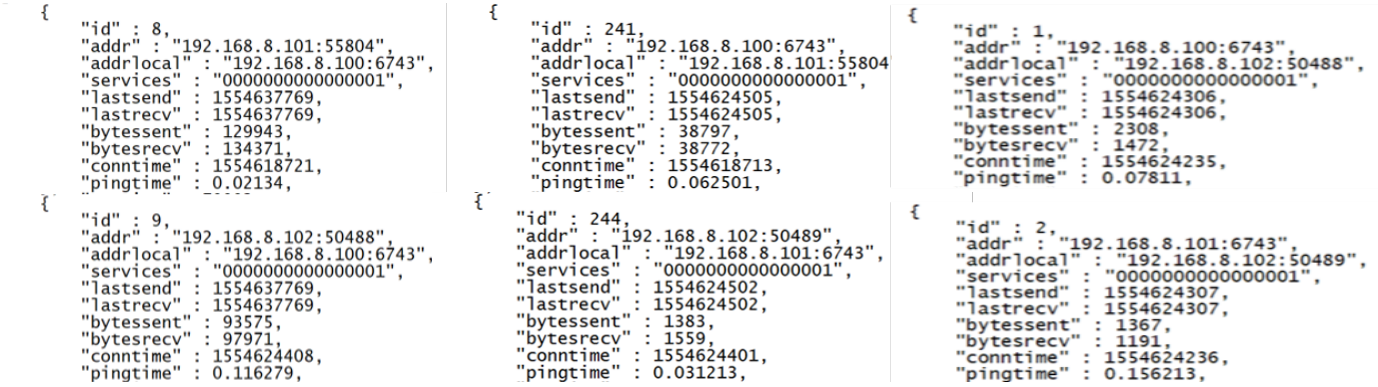
node 1 as compared to the ping time of connected node 2. It is very interesting to note that the ping time from connected node 1 and 2 to the seed node varies and is recorded as 0.062sec and 0.078sec respectively while from seed node it is 0.02sec and 0.11sec for connected node 1 and 2 respectively. The above data has been queried when no transaction was being carried out among the nodes. This data may change depending upon flux of transactions and upon contribution from each node. However, *pingtime* is an important parameter and indicates possibility of a successful transaction malleability attack by exploiting the difference in network latency encountered by different nodes.

Furthermore, in order to establish the experimentation scenario, a population of one thousand voters was created on the blockchain using the blockchain API. As our setup is a permissioned blockchain, voters' addresses were granted permissions to receive a single voting token as an asset and then later on utilizing the same voting asset by transferring it to the candidate's account. The process of granting permission for sending and receiving voting tokens is presented in Algorithm 1.

Algorithm 1 Granting Voting Rights

```
1: procedure VotingRightsAssignment(VotingChain, VoterList)
2:     BufferedReader ←VotingChain
3:     while BufferedReader ≠ EOF do
4:         VoterAddress ← BufferedReader[Counter]
5:         TransactionID ← AssgnVotingRights(VoterAddr, VotChain)
6:         Counter ← Counter + 1
7:     end while
8:     return TransactionID
9: end procedure
```

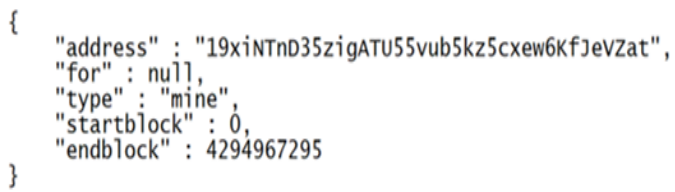
After assigning permission to voting tokens, these tokens are allocated to respective voters by the seed node which represents a voting regulatory body in the real world. The process adopted to achieve this is presented in Algorithm 2 whereas Fig 9 shows the output for the address and permission of this account as visible at the seed node. Furthermore, Fig 10 shows a sample status of the candidates for their rights.



```
{
    "id" : 8,
    "addr" : "192.168.8.101:55804",
    "addrlocal" : "192.168.8.100:6743",
    "services" : "0000000000000001",
    "lastsend" : 1554637769,
    "lastrecv" : 1554637769,
    "bytessent" : 129943,
    "bytesrecv" : 134371,
    "conntime" : 1554618721,
    "pingtime" : 0.02134,
{
    "id" : 9,
    "addr" : "192.168.8.102:50488",
    "addrlocal" : "192.168.8.100:6743",
    "services" : "0000000000000001",
    "lastsend" : 1554637769,
    "lastrecv" : 1554637769,
    "bytessent" : 93575,
    "bytesrecv" : 97971,
    "conntime" : 1554624408,
    "pingtime" : 0.116279,
{
    "id" : 241,
    "addr" : "192.168.8.100:6743",
    "addrlocal" : "192.168.8.101:55804",
    "services" : "0000000000000001",
    "lastsend" : 1554624505,
    "lastrecv" : 1554624505,
    "bytessent" : 38797,
    "bytesrecv" : 38772,
    "conntime" : 1554618713,
    "pingtime" : 0.062501,
{
    "id" : 244,
    "addr" : "192.168.8.102:50489",
    "addrlocal" : "192.168.8.101:6743",
    "services" : "0000000000000001",
    "lastsend" : 1554624502,
    "lastrecv" : 1554624502,
    "bytessent" : 1383,
    "bytesrecv" : 1559,
    "conntime" : 1554624401,
    "pingtime" : 0.031213,
{
    "id" : 1,
    "addr" : "192.168.8.100:6743",
    "addrlocal" : "192.168.8.102:50488",
    "services" : "0000000000000001",
    "lastsend" : 1554624306,
    "lastrecv" : 1554624306,
    "bytessent" : 2308,
    "bytesrecv" : 1472,
    "conntime" : 1554624235,
    "pingtime" : 0.07811,
{
    "id" : 2,
    "addr" : "192.168.8.101:6743",
    "addrlocal" : "192.168.8.102:50489",
    "services" : "0000000000000001",
    "lastsend" : 1554624307,
    "lastrecv" : 1554624307,
    "bytessent" : 1367,
    "bytesrecv" : 1191,
    "conntime" : 1554624236,
    "pingtime" : 0.156213,
```


(a) Connected nodes at seed node. (b) Peer info. at connected node 1. (c) Peer info. at connected node 2.

Figure 8: Node information for seed and connected nodes.



```
{
    "address" : "19xiNTnD35zigATU55vub5kz5cxew6KfJeVZat",
    "for" : null,
    "type" : "mine",
    "startblock" : 0,
    "endblock" : 4294967295
}
```


Figure 9: Vote assigning address as a miner.

Algorithm 2 Voting Asset Assignment

```
1: procedure VotingTokenAssignment(VotingChain, VoterList)
2:     VoteAssetObj ←OnChainVotingAsset
3:     BufferedReader ←VotingChain
4:     while BufferedReader ≠ EOF do
5:         VoterAddress ← BufferedReader[Counter]
6:         TransactionID ←
           SendVotingToken(AuthorizedWalletAddress, VoterAddress,
           VoteAssetObj[Token])
7:         Counter ← Counter + 1
8:     end while
9:     return TransactionID
10: end procedure
```

```
[
    {
        "address" : "1CCvcFES4H7HfHktbU9Fn59szHbN2UoCwpF6sf",
        "for" : null,
        "type" : "connect",
        "startblock" : 0,
        "endblock" : 4294967295
    },
    {
        "address" : "1CCvcFES4H7HfHktbU9Fn59szHbN2UoCwpF6sf",
        "for" : null,
        "type" : "send",
        "startblock" : 0,
        "endblock" : 4294967295
    },
    {
        "address" : "1CCvcFES4H7HfHktbU9Fn59szHbN2UoCwpF6sf",
        "for" : null,
        "type" : "receive",
        "startblock" : 0,
        "endblock" : 4294967295
    }
]
```


Figure 10: Sample candidate rights status.

7.2. Experimentation

The experimentation performed to execute the transaction malleability attack has been divided into two cases as explained below:

- Case 1: Experimentation setup with no explicit efforts to introduce delays to facilitate malleable transaction and works on taking chance to get the malleable transaction mined into the block. Therefore, transaction malleability is successful due to computation power of attacker's machine, network delay and due to miner's choice of transaction to be mined.
- Case 2: In contrast to case 1, this setup induces delays to observe the impact of software-level delays.

7.2.1. Case 1:

In this case, an honest transaction was sent from connected node 1 where a voter moves their voting token to the candidate's address. Two other parallel running voting client applications were used to compete with this honest transaction. As mentioned above, the connected node 1(which is sending honest transactions) is slightly weaker in terms of computational power than the attacker nodes to increase the probability of having a successful transaction malleability attack. The success of this attack depends upon the computational strength of machine, propagation time of the network, and the node connectivity with the seed node which may vary in accordance with the load on the network.

At the start of experiments, a series of honest transactions are issued from an honest node. As soon as the first transaction attempts to move the vote from the voter's address to the candidate's address, the attacker nodes attempt to get the malleable transaction added into the block. Table 3 below shows the important fields and values of the honest and its malleable transaction in this experiment.

S. No.	Voter Address	Candidate Address	Asset Name	Metadata
01	1K4FBpduhmwZKNTvHV3j 4sDPKWGLQh4DiMyvFH	1CCvcFES4H7HfHktbU9 Fn59szHbN2UoCwpF6sf	VotingAsset Serial2	Honest Tx for No. "N" at (PC2)
02	1K4FBpduhmwZKNTvHV3j4s DPKWGLQh4DiMyvFH	1CCvcFES4H7HfHktbU9 Fn59szHbN2UoCwpF6sf	VotingAsset Serial2	Malleable Tx for No. "N" at (PC3)

Table 3: Honest / malleable version of transaction with successful attack.

It is evident from Table 3 that the malleable version of the transaction was created by manipulating the contents of the metadata field of the transaction. Using the honest and malleable transaction specifications above, the malleable transaction was able to successfully challenge and win race against its honest transaction.

Fig 11.a and Fig 11.b shows the acceptance and rejection of transaction from connected nodes 1 and 2 respectively.

```
Voting Transaction is being sent at 2019-04-21T18:28:05.816 for voter
1K4FBpduhmwZKNTvHV3j4sDPKWRQh4DiMyvFH
TxID: 1561bf202aafe4edb37d9bd832eeebf45de74a89cf3f139f37cd48e9cfc2ae281
TxNo 1 at 2019-04-21T18:28:15.690
```

(a) Malleable transaction acceptance from connected node 2.

```
Voting Transaction is being sent at 2019-04-21T18:28:05.333 for Voter
1K4F8pduhmwZKNTvHV3j4sDPKWGLQh4DiMyvFH
ErrorMultichainException [
                        object=code :-6,0,
                        reason=message : Insufficient funds,
                        message=null,
                        cause=null
]
```

(b) Honest transaction rejection from connected node 1.

Figure 11: Structure for honest and malleable transactions.

Fig 11 presents the output of voting client program when the malleable and honest transaction was attempted from the honest and attacker node. It is evident that although the honest transaction from connected node 1 was sent before the malleable transaction, the malleable transaction was mined mainly due to network delay, block propagation speed to the consensus node, computational power of these two different nodes and by being chosen by the miner before its honest counterpart. Fig 11 also shows the time elapsed of acceptance and rejection of malleable and honest transaction respectively with respect to when it was released from the client and when it responded to the client in case of a successful or unsuccessful transaction. It is also evident from Fig 11, the honest transaction was picked up by the miner well after its malleable transaction was mined as the client software received the confirmation response i.e. almost after 10 seconds from the time when it was sent.

Since the blockchain network used in this experiment is private, the timing of the malleable transaction plays a key role here. Essentially, the greater the difference between the sending of honest and malleable transaction, the lesser are the chances of successfully carrying out the transaction malleability attack. Fig 11.b shows that the honest transaction was sent at 18:28:05.333 from connected node 1 and immediately after that, its malleable transaction was sent from connected node at 18:28:05.816 (Fig 11.a) which was added to the block. The difference in the issuance time of honest and malleable transaction was kept low in order to carry out a successful transaction malleability attack. This difference may vary depending upon the number of unconfirmed transactions in the pool, number of miners in the blockchain, size and bandwidth of the network.

Analysis of Transaction Execution: Fig 14 confirms the transaction hash of the malleable transaction in the blockchain which was observed by querying from the seed node. Here, the addresses of voter and candidate can be seen along with the transaction ID which was generated when the malleable transaction was accepted into the blockchain. Similarly there are some other interesting statistics in Fig 14. For instance, the time when the node received the malleable transaction and the time when that malleable transaction was added to the local wallet of the node by the parameters *timereceived* and *time* respectively. This time may vary across nodes as it shows the local time of the node but this is not the case in this experiment as all the nodes were synchronized through the Internet time synchronization. Another important parameter here is the blocktime which requires every node to be agreed upon which helps develop the consensus blockchain. The values for all these time specific parameters are shown as UNIX timestamps. Table 4 shows these timestamps in human readable format.

Table 4 shows that the malleable transaction almost took 5 seconds to be mined into the block. A very interesting parameter *data* is also shown in Fig 14 in the hexademical format which is computed from the contents of metadata used not only for creating the malleable transaction but also a confirmation for which transaction (honest or malleable) was mined into the block. When the data is converted into text then the output of the text, *Malleable Tx for No. 1 at (PC2)* also confirmed that the malleable transaction was mined into the block demonstrating successful execution of transaction malleability attack.

Discussion and Analysis of Results: Through the

```
{
    "balance" : {
        "amount" : 0,
        "assets" : [
        ]
    },
    "myaddresses" : [
        "1CCvcFES4H7HfHktbU9Fn59szHbN2UoCwpF6sf",
        "1K4FBpduhmwZKNTvHV3j4sDPKWGLQh4DiMyvFH"
    ],
    "addresses" : [
    ],
    "permissions" : [
    ],
    "items" : [
    ],
    "data" : [
        "4d616c6c6561626c6520547820666f72204e6f203120617420285043329"
    ],
    "confirmations" : 23,
    "blockhash" : "002c81830c37ea1a978ef87449290310fe5d9000e27408eddd167e9af517eccf",
    "blockindex" : 1,
    "blocktime" : 1555853292,
    "txid" : "1561bf202aafe4edb37d9bd832eeebf45de74a89cf3f139f37cd48e9cfc2ae28",
    "valid" : true,
    "time" : 1555853287,
    "timereceived" : 1555853287
}
```


Figure 12: Malleable transaction in the wallet of seed node.

Malleable Tx ID	Time Received	Time Added to Wallet	Block Time
1561bf202aafe4edb37d9bd832eeebf45de74a89cf3f139f37cd 48e9cfc2ae28	6:28:07 PM	6:28:07 PM	6:28:12 PM

Table 4: Time-wise status of malleable transaction

experimentation, it is evident that the probability of getting malleable transaction mined into the block is lower as compared to getting an honest transaction into the block. In this experiment, various attempts were made to carry out a successful attack with different delays between an honest and a malleable transaction and it was identified that the chances for carrying out a successful transaction malleability attack were prominent when the time difference between sending an honest and its malleable transaction was very low.

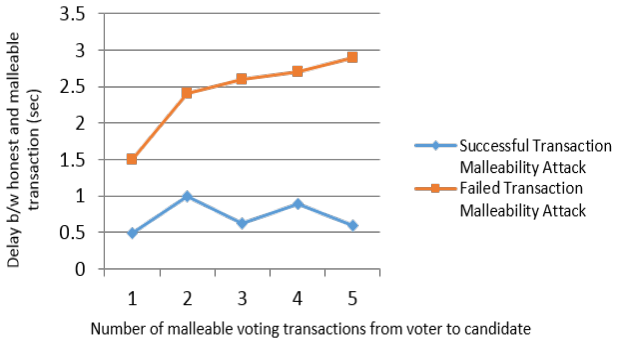


Figure 13: Successful vs failed transaction malleability attack.

It is evident from Fig 13, within the experiments performed, the likelihood of a successful transaction malleability attack is high for a time window from 0.483 to 0.995 seconds whereas a delay of 1.5 seconds or more has very low likelihood of a successful transaction

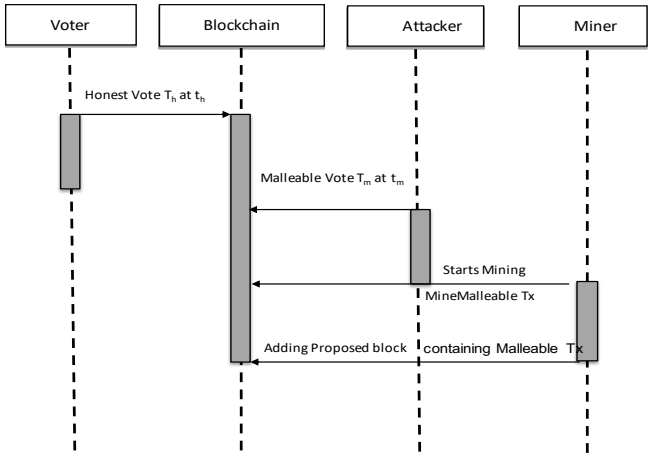
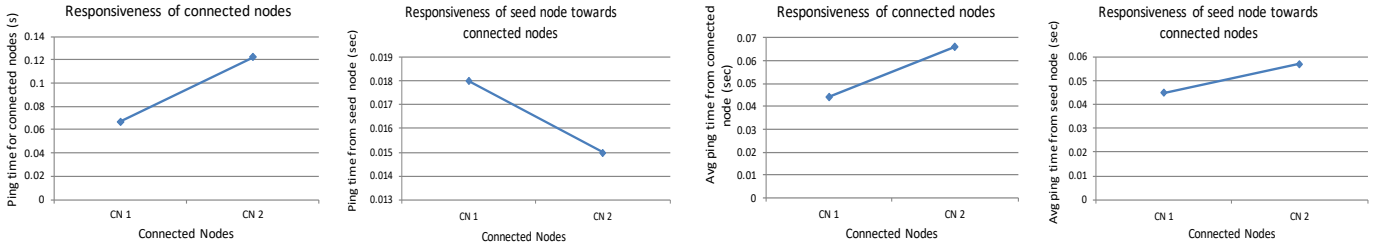


Figure 14: Sequence of events for a successful transaction malleability attack.

malleability. However, there are other factors such as connectivity strength (presented in Fig 15 which play an important role in the success of this attack.

Fig 15 describe the responsiveness in terms of ping time between seed/root and connected nodes. It specifically shows the state of the network at the moment when the blockchain network was under transaction malleability attack. In this case, the connectivity time between the seed node and connected node 2 is shorter as compared to the same for connected node 1. Further, the process of transaction formation is quicker in the attacker



(a) Connectivity strength from connected to seed node. (b) Connectivity strength from seed to connected node. (c) Average connectivity strength from connected to seed node. (d) Connectivity strength from seed to connected node (just before attack).

Figure 15: Connectivity strength for seed and connected nodes across different settings.

node due to its computational strength. Another interesting fact is that since the ping time from root node to attacker node (connected node 2) is lesser than the honest node, the building up of the consensus blockchain and its synchronization process is expected to occur in quicker time for connected node 2 (attacker node) than in connected node 1 (honest node). Although these are some of the factors which can influence the success of transaction malleability attack, the malleable transaction was received and picked up by the miner earlier than its respective honest transaction. The ping time in the network is a variable parameter which keeps changing its value at different time instances. Fig 15.b shows the state of the network with respect to its ping time just after the attack was successful. At this instance, it can be observed that the connection between seed node and connected node 2 (attacker's node) was slightly better than that between seed node and connected node 1 (honest node).

In order to understand an overall consistent behavior of the system, five readings were taken at different occasions of time as presented in Table 5. As evident from Fig 15.c and Fig 15.d, it can be concluded that the responsiveness of the victim's node was better than the attacker's node except the cases at row number 4 of Table 5. The data was also recorded for the round trip time from seed node to honest and attacker's node and in this record keeping activity row number 3 contained the only captured data where the round trip from seed to connected node 2 (attacker's node) took lesser time than the round trip from seed node to honest node. Therefore, these factors have caused the attack to succeed as the reading at 0.015 and 0.018 was observed to successfully execute a transaction malleability attack causing the malleable transaction to enter into the consensus blockchain before the honest transaction.

Fig 16 show the individual readings recorded at five different occurrences to show the state of network communication through ping time between different nodes. It specifically shows the overall state of the network to determine and assess a general behaviour of the network over a period of time while it was being used for experimentation. These measurements reflect the overall connectivity among the nodes of the network irrespective of the success or failure of attack. Here, connected node 2 (attacker's node) was witnessed to be more responsive to the root node which facilitated the transaction malleability attack as such an attack can only be successful when there is a delay in mining honest transaction. Further, Fig 16.g presents the round trip time to analyse the process of consensus and data synchronization so as to understand its impact on transaction receipt and update of local blockchain. This is important as this activity requires a complete round of communication from root node to the connected nodes. Therefore, the sooner the synchronization occurs, the sooner the nodes agree upon the consensus state of the blockchain to maintain a consistent state of the blockchain. Fig 16 displays the relationship of the round trip time between seed node and its associated connected nodes including honest and attacker nodes.

Algorithm 3 TM Attack

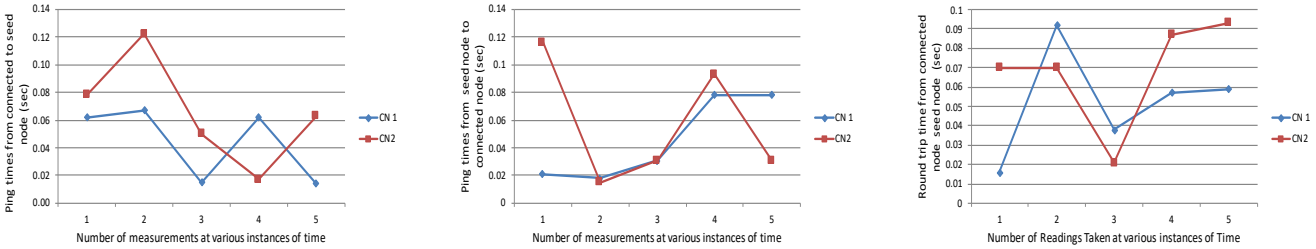
procedure TransactionMalleability($VotingChainWallet$, $VoterList$, $VotingChain$)
  $VoteAssetObj$ ←OnChainVotingAsset
  $BufferedReader$ ←VotingChain
  while $BufferedReader \neq EOF$ **do**
    $VoterAddress \leftarrow BufferedReader[Counter]$
    $HonestTransaction \leftarrow TransactionFormation(HonestClientNode, VoterAddress, CandidateAdress, VotingToken, HonestMetaData)$
    $MalleableTransaction \leftarrow TransactionFormation(AttackerClientNode, VoterAddress, CandidateAdress, VotingToken, MalleableMetaData)$
    $RandomDelay \leftarrow Random(Number)$ ▷ At Honest Node
    Sleep(RandomDelay) ▷ At Honest Node
    $TransactionID \leftarrow SendVote(HonestTransaction)$ ▷ At Honest Node
    $RandomDelay \leftarrow Random(Number)$ ▷ At Attacker Node
    Sleep(RandomDelay) ▷ At Attacker Node
    $TransactionID \leftarrow SendVote(MalleableTransaction)$ ▷ At Attacker Node
    $Counter \leftarrow Counter + 1$
  end while
  return $SuccessfullyExecutedTransaction$
end procedure

7.2.2. Case 2:

This set of experiments was performed to analyze the impact of delayed transaction at software level in contrast to the network level delay in case 1. In this scenario, an honest transaction was created and sent

S. No.	Ping Time from CN1 to Seed Node(s)	Ping Time from CN2 to Seed Node(s)	Ping Time from Seed Node to CN1 (s)	Ping Time from Seed Node to CN2 (s)	RTT from CN1 to Seed Node (s)	RTT from CN2 to Seed Node (s)
1	0.062	0.078	0.021	0.116	0.016	0.070
2	0.067	0.122	0.018	0.015	0.092	0.070
3	0.015	0.050	0.031	0.031	0.038	0.021
4	0.062	0.017	0.078	0.093	0.057	0.087
5	0.014	0.063	0.078	0.031	0.059	0.093

Table 5: Connectivity Strength among Nodes



(a) Ping time at connected node for seed node across different time instances.

(b) Ping time at seed node for connected node across different time instances.

(c) Round trip time from connected node to seed node across different time instances.

Figure 16: Round trip time from connected node to seed node across different time instances.

earlier than its corresponding malleable transaction but with an explicit delay induced at application level through the client software of the honest node. The delay was varied to analyze its impact on the blockchain mining process and its block generation rate. For instance, if the delay lies within the window of issuing the next block (15 seconds in this case), then both the honest and malleable transactions may become the contender for the same block. However, it also depends upon the already elapsed time after the arrival of latest block i.e. if the honest transaction comes at 5 seconds after the addition of latest block and its malleable transaction comes at 8 seconds after the creation of the same block then in this case honest transaction will have 10 seconds and its malleable transaction will have 7 seconds to get itself into the next block. Similarly, there may be another case when the malleable transaction reaches with the same delay of two seconds, however, due to this delay the honest transaction may be mined with the previous block and this 2 seconds delay may have caused one block to enter into the blockchain and the malleable transaction just came 2 seconds after the addition of previous block(after the average 15 seconds windows of block). Fig 14 shows a typical sequence for the occurrences of events for a successful transaction malleability attack within this case.

As part of this experiment, a number of iterations were run simultaneously from honest and attacker's node to enable multiple malleable transactions to get into the blockchain disregarding if previous transactions are successful in achieving transaction malleability. The process used to achieve this is explained by the pseudocode in Algorithm 3. As explained in Algorithm 3, if an honest transaction is sent earlier than its corresponding malleable transaction but failed to mine due to the delay between the release of honest and malleable transaction is less than the random delay that honest transaction would face before being forwarded to blockchain as compared to the delay which malleable transaction would face before being forwarded to the blockchain.

Although many iterations were run to analyze feasibility of a successful transaction malleability attack, Table 6 shows some of transaction's arrival time and the time when the transaction was mined along with the delay in between the transactions. Table 6 also presents the time when the malleable transaction became part of the blockchain, its transaction ID along with the time of confirmation from the e-voting blockchain.

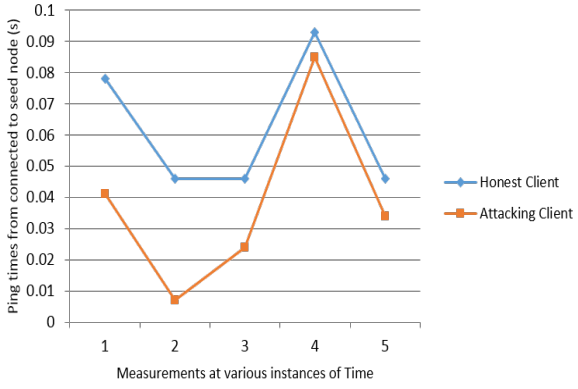


Figure 17: Connectivity strength from connected to seed node.

Discussion and Analysis of Results: As demonstrated by Fig 17, the ping time for honest client position was slightly better than the attacker. In order to understand the overall behavior of the system, Fig 18.a illustrates the average response of ping time for the

S. No.	Honest Tx Release time (hh:mm:ss.SSS)	Malleable Tx Release time (hh:mm:ss.SSS)	Random Delay for Honest Node (s)	Random Delay for Attacker Node (s)	Successful Transaction	Successful Transaction ID
1	03:53:43.741	03:53:51.927	7	3	Honest	6d22978274d0917c43eacd9215303a06c ecd18e6a0d57962268843b536217db6
2	03:53:45.085	03:53:53.131	0	2	Honest	831d5f58da8a0d76dcbeff57ff761c5f 3797c2e5cc478328fc31baf79e900a09
3	03:53:53.163	03:53:57.162	8	3	Malleable	5bda0788da20534051d945ec45a860ab b18a04d36da2151c07fe0ea93b3e6f83
4	03:54:01.226	03:54:02.214	8	5	Malleable	3414e0631ea1c498a57931af32e5f169 e3a343b533e36499fb84bae80980a738
5	03:54:03.304	03:54:03.272	2	2	Malleable	9e017987469853566cd1dcc8575a261a 2addfbede92e82667e7e92237a3d49a4

Table 6: Sample transactions for successful/failed transaction malleability.

S. No.	Successful TxID	VoterID	Mining Time (s)	Voter Number	Encoded Hexadecimal Meta data
1	6d22978274d0917c43eacd921 5303a06cecd18e6a0d57 962268843b536217db6	1K4FBpduhmwZKNTvHV3j 4sDPKWGLQh4DiMyvFH	7	1	486f6e6573742054782066f72204e6f 2e2031206174202850433129
2	831d5f58da8a0d76dcbeff57 ff761c5f3797c2e5cc4783 28fc31baf79e900a09	1JoxBh4ime2EcnarYmX wLgpvwMpNFhRDrJJXXZ	7	2	486f6e6573742054782066f722 04e6f2e203220617420285043312 9
3	5bda0788da20534051d945 ec45a860abb18a04d36da21 51c07fe0ea93b3e6f83	13aw1Y6hA2pmyyG3sNx k2EBgessvEuaLTECwAv	9	3	4d616c6c6561626c6520547820666f72204e6 f2e2033206174202852 6f6f74204e6f646529
4	3414e0631ea1c498a57931 af32e5f169e3a343b533e 36499fb84bae80980a738	1J3jHejHQtavKV7u3Q wSKg3TKFcCGDbJd1TtqK	4	4	4d616c6c6561626c6520547820666f72204e 6f2e2034206174202852 6f6f74204e6f646529
5	9e017987469853566cd1dcc 8575a261a2addfbede92e8266 7e7e92237a3d49a4	1RkifPykMCHY4ZMnq8 SU3eSJhsCnyhdrFpb5eB	2	5	4d616c6c6561626c6520547820666f72204e 6f2e2035206174202852 6f6f74204e6f646529

Table 7: Metadata for successful/failed transaction malleability.

honest and attacking client of this experiment. It can be concluded through analysis of Fig 17 and Fig 18.a that the honest client performed better in terms of connectivity. The other factor which was considered is that the honest client exhibits comparatively weaker computational node than the malicious client. Consequently, the malicious node can potentially generate higher number of malleable transactions so as to increase the likelihood of getting mined into the blockchain. There may be a possibility in a real world scenario where one miner picks up the honest transaction and the other picks up its equivalent malleable transaction. In this case, the transaction that gets mined into the block first is likely to be added to the consensus blockchain upon agreeing by all the nodes of the network.

Fig 18.b and c explain the difference of responses between honest and malleable transaction against varying delay. These figures show the state of the blockchain of a single activity when both the honest and attacking clients send concurrent bulk transactions with a randomly generated delay to give their transactions a chance of getting mined earlier than their respective semantically similar transaction. Further, as presented in Table 6 and Fig 18.b and c, the transaction with relatively lesser delay is usually the winner. For instance, row number 5 of Table 6 presents a very interesting scenario when both transactions (honest and malleable) have equally same delay, but the succeeding transaction is the malleable transaction due to a slight difference of release time. This is due to the fact the attacker node was able to process and send transaction quicker than the honest node. Even an honest node forms the transaction before the attacker‘s node, the attacker may still make an impact by processing to release it earlier. Although the difference is very small (32 milliseconds), however, it caused the malleable transaction to be added to the block leading to a successful attack.

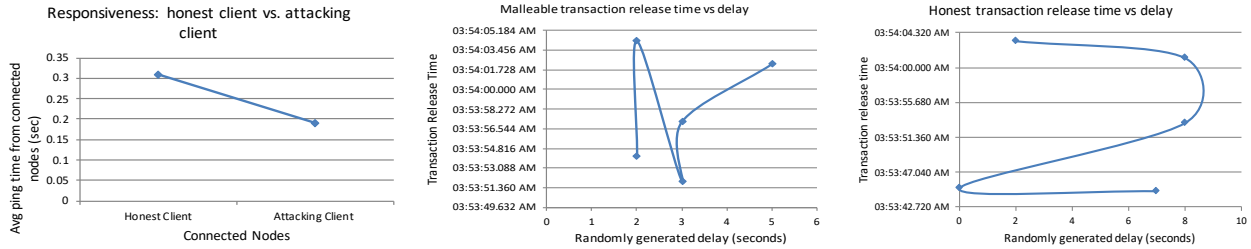
8. Protection against transaction malleability attack

In this paper, we have successfully demonstrated the feasibility of transaction malleability attack within blockchain-based systems. Therefore, we highlight a gap in current literature to protect against such attacks. Continuing from this work, our current and future research is focused at developing innovate solutions to enhance security of blockchain-based solutions. In particular, we envisage investigating a novel provenance-based architecture framework to develop a solution for countering the transaction malleability attack at the blockchain core.

Specifically, the provenance-based scheme will aim to block the transaction malleability attack by creating an additional layer of provenance at the top of blockchain. This layer will keep the origin of actual transactions which became the part of the blockchain network first by utilizing the same unspent unit of asset (which is a single

Successful Malleable Tx ID	Time Received	Time Added to Wallet	Block Time	Decoded Hex Metadata
5bda0788da20534051d945ec45a860abb18a04d36da215 1c07fe0ea93b3e6f83	3:53:57 AM	3:53:57 AM	3:54:06 AM	Malleable Tx for No. 3 at root node
3414e0631ea1c498a57931af32e5f169e3a343b533e36 499fb84bae80980a738	3:54:02 AM	3:54:02 AM	3:54:06 AM	Malleable Tx for No. 4 at root node
9e017987469853566cd1dcc8575a261a2addfbede92e8 2667e7e92237a3d49a4	3:54:04 AM	3:54:04 AM	3:54:06 AM	Malleable Tx for No. 5 at root node

Table 8: Selected malleable transactions for successful attack.



(a) Responsiveness of honest vs malicious client.

(b) Impact of delay on malleable transaction.

(c) Impact of delay on honest transaction.

Figure 18: Impact of delay on malleable and honest transactions.

vote cast according to the voting model in Figure 2). By doing so, this layer will not only reject the next incoming semantically similar transactions to be a part of the consensus blockchain but it will also not require to go through the compute-hungry process of traditional verification which requires tracing the origin of the particular unspent unit of asset (a vote in the context of the e-voting application presented here) from its genesis transaction.

Furthermore, we envisage evaluating the provenance-based scheme to counter transaction malleability attack in the proposed voting model against varying conditions. These include scenarios when the voting process is running normally and when the voters are issuing bulk voting transactions concurrently (with different block generate rates, mining diversity). Such rigorous evaluation will also help research community to analyse the effectiveness of the provenance-based solution with respect to the impact of transaction malleability attack when the blockchain is subjected to an attack using a attack strategy similar to that used in Mt. Gox exchange attack.

9. Conclusion and future work

Blockchain has inspired a diverse range of applications, which seek to leverage its benefits such as tamper-proof ledger and transparent access to information. However, transaction malleability is one of potential threats to blockchain which can lead to double-spending attacks. This paper has presented an empirical analysis of the transaction malleability threat to blockchain identifying the role of parameters such as network delay, block generation rate and software-induced delays to achieve a successful transaction malleability attack. The paper has used a real-world blockchain test-bed for its experiments hosting an e-voting application, however, as the transaction malleability attack targets blockchain fabric, the outcomes of this research also impacts blockchain-based applications in other domains such as logistics and healthcare. Our current and future work focuses on developing mechanisms and methods which can mitigate against transaction malleability attack demonstrated here.

References

[1] M. Andrychowicz, S. Dziembowski, D. Malinowski, and Ł. Mazurek, "On the malleability of bitcoin transactions," in *Financial Cryptography and Data Security*, M. Brenner, N. Christin, B. Johnson, and K. Rohloff, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 1–18.

[2] K. M. Khan, J. Arshad, and M. M. Khan, "Secure digital voting system based on blockchain technology," *Int. J. Electron. Gov. Res.*, vol. 14, no. 1, pp. 53–62, Jan. 2018. [Online]. Available: https://doi.org/10.4018/IJEGR.2018010103

[3] N. Nizamuddin, K. Salah, M. A. Azad, J. Arshad, and M. Rehman, "Decentralized document version control using ethereum blockchain and ipfs," *Computers Electrical Engineering*, vol. 76, pp. 183 – 197, 2019. [Online]. Available: https://bit.ly/2NMvKHN

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[5] D. Chaum, "Secret-ballot receipts: True voter-verifiable elections," *IEEE Security Privacy*, vol. 2, no. 1, pp. 38–47, 2004.

[6] A. Barnes, C. Brake, and T. Perry, "Digital voting with the use of blockchain technology," 2016, the Economist Competition on Blockchain based e-voting. [Online]. Available: https://www.economist.com/sites/default/files/plymouth.pdf

[7] H. Baldersheim and J. Saglie, "Internet voting in norway 2011: Democratic and organisational experiences," in *The 4th International Conference on Democracy as Idea and Practice*, 2013.

[8] E. M. of Foreign Affairs, "Estonian internet voting system," 2019.

[9] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, 2017. [Online]. Available: https://bit.ly/2q14WLW

[10] I.-C. Lin and T.-C. Liao, "A survey of blockchain security issues and challenges," *I. J. Network Security*, vol. 19, pp. 653–659, 2017.

[11] Multichain. Open platform for blockchain applications. [Online]. Available: www.multichain.com

[12] Y. Lu, "Blockchain: A survey on functions, applications and open issues," *Journal of Industrial Integration and Management*, vol. 03, 08 2018.

[13] B. Asolo. Txid (transaction identifier) explained. [Online]. Available: https://www.mycryptopedia.com/txid-transaction-identifier-explained

[14] C. Decker and R. Wattenhofer, "Bitcoin transaction malleability and mtgox," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 313–326.

[15] J. P. J. S. Kadam, M., "Double spending prevention in bitcoins network," *International Journal of Computer Engineering and Applications*, 2015.

[16] G. Karame, "On the security and scalability of bitcoin's blockchain," in *The 2016 ACM SIGSAC Conference*, 10 2016, pp. 1861–1862.

[17] A. P. Ozisik and B. N. Levine, "An explanation of nakamoto's analysis of double-spend attacks," *CoRR*, vol. abs/1701.03977, 2017. [Online]. Available: http://arxiv.org/abs/1701.03977

[18] C. Pérez-Solà, S. Delgado-Segura, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Double-spending prevention for bitcoin zero-confirmation transactions," *International Journal of Information Security*, vol. 18, no. 4, pp. 451–463, Aug 2019. [Online]. Available: https://doi.org/10.1007/s10207-018-0422-4

[19] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 129–144.

[20] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 251–260.

[21] M. Vasek, M. Thornton, and T. Moore, "Empirical analysis of denial-of-service attacks in the bitcoin ecosystem," in *International conference on financial cryptography and data security*. Springer, 2014, pp. 57–71.

[22] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the delivery of blocks and transactions in bitcoin," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 692–705.

[23] J. Heusser, "Sat solving-an alternative to brute force bitcoin mining," 2013.

[24] H. Finney, "Best practice for fast transaction acceptance-how high is the risk," 2011.

[25] J. A. Kroll, I. C. Davey, and E. W. Felten, "The economics of bitcoin mining, or bitcoin in the presence of adversaries," in *Proceedings of WEIS*, vol. 2013, 2013, p. 11.

[26] J. Bonneau, "Why buy when you can rent?" in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 19–26.

[27] Corbixgwelt. Timejacking and bitcoin. [Online]. Available: http://culubas. blogspot.de/ 2011/ 05/timejacking-bitcoin 802.html/

[28] N. T. Courtois and L. Bahack, "On subversive miner strategies and block withholding attack in bitcoin digital currency," *arXiv preprint arXiv:1402.1718*, 2014.

[29] M. Rosenfeld, "Analysis of bitcoin pooled mining reward systems," *arXiv preprint arXiv:1112.4980*, 2011.

[30] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 515–532.

[31] P. Wuille, "Disclosure: consensus bug indirectly solved by bip66," 2015. [Online]. Available: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2015-July/009697.html

[32] M. Rosenfeld, "Analysis of hashrate-based double spending," *CoRR*, vol. abs/1402.2009, 2014. [Online]. Available: http://arxiv.org/abs/1402.2009

[33] U. Rajput, F. Abbas, and H. Oh, "A solution towards eliminating transaction malleability in bitcoin," *JIPS*, vol. 14, pp. 837–850, 2018.

[34] H. Schoenfeld. Malleability attack and why it matters. [Online]. Available: https://bit.ly/348Rbti

[35] N. Hourt, "Blockchain technology in online voting," 2017. [Online]. Available: https://followmyvote.com/online-voting-technology/blockchain-technology/

[36] M. Rockwell, "Bitcongress—process for blockchain voting & law," 2017.

[37] Z. Zhao and T.-H. H. Chan, "How to vote privately using bitcoin," in *International Conference on Information and Communications Security*. Springer, 2015, pp. 82–96.

[38] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 397–411.

[39] M. H. Ibrahim, "Securecoin: A robust secure and efficient protocol for anonymous bitcoin ecosystem." *IJ Network Security*, vol. 19, no. 2, pp. 295–312, 2017.

[40] M. Chaieb, S. Yousfi, P. Lafourcade, and R. Robbana, "Verify-your-vote: A verifiable blockchain-based online voting protocol," in *European, Mediterranean, and Middle Eastern Conference on Information Systems*. Springer, 2018, pp. 16–30.

[41] M. Chaieb, M. Koscina, S. Yousfi, P. Lafourcade, and R. Robbana, "Dabsters: A privacy preserving e-voting protocol for permissioned blockchain," in *International Colloquium on Theoretical Aspects of Computing*. Springer, 2019, pp. 292–312.

[42] K. M. Khan, J. Arshad, and M. M. Khan, "Simulation of transaction malleability attack for blockchain-based e-voting," *Computers and Electrical Engineering*, 2020.

[43] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and cryptocurrency technologies: A comprehensive introduction*. Princeton University Press, 2016.

[44] M. Rosenfeld, "Analysis of hashrate-based double spending," 02 2014.

[45] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

[46] D. Khader, B. Smyth, P. Ryan, and F. Hao, "A fair and robust voting system by broadcast," *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI)*, vol. 205, 01 2012.

[47] F. Hao, M. N. Kreeger, B. Randell, D. Clarke, S. F. Shahandashti, and P. H. J. Lee, "Every vote counts: Ensuring integrity in large-scale electronic voting," in *2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 14)*. USENIX Association, 2014.

[48] A. Kiayias and M. Yung, "Self-tallying elections and perfect ballot secrecy," in *Public Key Cryptography*, D. Naccache and P. Paillier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 141–158.

[49] K. M. Khan, J. Arshad, and M. M. Khan, "Investigating performance constraints for blockchain based secure e-voting system," *Future Generation Computer Systems*, vol. 105, pp. 13 – 26, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X19310805