Computers and Electrical Engineering 83 (2020) 106583

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Simulation of transaction malleability attack for 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

ARTICLE INFO

Article history:
Received 29 June 2019
Revised 6 November 2019
Accepted 12 February 2020
Available online 25 February 2020

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

ABSTRACT

Blockchain has been adopted to address significant challenges, such as trust in diverse domains, including voting, logistics and finance. However, transaction malleability has been identified as a threat for blockchain, which can potentially lead to an inconsistent state that can result in further attacks such as double-spending. In this context, this paper is focused on investigating the feasibility of transaction malleability within a typical blockchain application aiming to identify scenarios that may lead to a successful transaction malleability attack. Our objective in doing so is to highlight conditions which cause such attack to facilitate the development of protection mechanisms for them. Specifically, this paper presents a successful simulation of transaction malleability attack within the context of blockchain-based electronic voting. The evaluation has identified the impact of parameters, such as network delay and block generation rate in achieving a successful transaction malleability attack, which highlights future directions of research.

1. Introduction

The fundamental concept of the blockchain data structure is similar to a linked list. It is shared among all the nodes of the network where each node keeps its local copy of all the blocks (associated with the longest chain) starting from its genesis block [1]. Blockchain technology has introduced a new model of application development primarily based on the successful implementation of the data structure within the Bitcoin application. Recently, many real-world applications have been developed in diverse domains, such as the Internet of Things [2], artificial intelligence [3] and e-document management [4]. These applications leverage benefits of blockchain technology due to its self-cryptographic validation structure among transactions (through hashes), and public availability of distributed ledger of transaction-records in a peer-to-peer network. Creating a chain of blocks connected by cryptographic constructs (hashes) makes it very difficult to tamper the records, as it would cost the rework from the genesis to the latest transaction in blocks as illustrated by Nakamoto [5].

The above-defined characteristics of blockchain have attracted significant interest from a wide range of applications to leverage benefits of blockchain technology, especially where trust is of fundamental importance. Consequently, blockchain has enabled the development of a decentralized trustworthy environment for diverse application domains, from cryptocurrency to electronic voting system, banking and finance industry to supply-chain management, and from health care to

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Prof. Khaled Salah.

* Corresponding author.
E-mail address: junaid.arshad@bcu.ac.uk (J. Arshad).

https://doi.org/10.1016/j.compeleceng.2020.106583
0045-7906/

blockchain-based identities, such as passport and e-residency. Within this context, leveraging cryptographic fundamentals underpinning blockchain technology, the cost of illegitimate record tampering is higher than reproducing these records [6] thereby achieving tamper-resistant auditing of information.

Although research around e-voting systems has matured in recent years, a major concern in such systems is the trustworthiness of the voting processes as well as the security of the e-voting software and its resilience against emerging threats. This raises several new technical concerns such as the security of governance process within such e-voting system, the security of the e-voting software against application-level threats, the protection mechanisms implemented to achieve secure, tamper-proof auditing. One of the limitations of e-voting systems including Direct Routing Electronic (DRE) voting is that it does not provide end to end (E2E) verifiability requirements set out in [7]. In E2E verifiable election system, the leverage is given with which it can be verified by the voter that their vote has been cast as per his will. The casting vote must have a part of the record which can be tallied later on as identified by Hao et al. [8].

Within this context, blockchain technology can be beneficial for implementing e-voting where it will be difficult to reuse the token (vote) twice, as in order to be a part of blockchain, all the nodes are required to be initialized from the same genesis block to add transactions into the blockchain which is synchronized among all peers. Therefore when this technology is adopted within e-voting, the foundations of blockchain can potentially increase the cost of double utilization of a token exponentially 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. Moreover, since blockchain technology is decentralized, every node will be running their local copy of the blockchain which may later be used for consensus as elaborated in [8,9].

This paper is focused on investigating challenges in the adoption of blockchain to achieve decentralized trustworthy applications, considering e-voting as an application scenario with particular focus on the public voting model. E-voting is one of the emerging applications of blockchain whereby researchers have aimed to leverage features, such as integrity, anonymity and non-repudiation, that are critical for a voting application. Use of blockchain to facilitate e-voting applications has recently received significant attention with efforts such as [10] leveraging blockchain technology to achieve secure and verifiable voting. However, transaction malleability has been identified as a threat for blockchain-based applications, which can lead to an inconsistent state potentially resulting in further attacks, such as double-spending. This paper, therefore, presents the outcome of our investigation into the feasibility of transaction malleability within blockchain-based e-voting. We aim to identify scenarios which may lead to a successful transaction malleability attack, thereby highlighting conditions causing such attack to facilitate the development of protection mechanisms for them.

Rest of the paper is organized as follows. Section 2 presents a thorough discussion around the transaction malleability attack in blockchain and its potential impact on different application domains, including e-voting. This is followed by a critical review of existing efforts concerning attack simulations within blockchain as well as our application in focus i.e. e-voting in Section 3. Attack model for transaction malleability is explained in Section 4, followed by a detailed discussion about the simulation of transaction malleability in Section 5. Experimentation results are presented in Section 6 followed by an in-depth analysis of these results in Section 7. Section 8 concludes the paper.

2. Transaction malleability attacks in blockchain

Although blockchain technology has attracted significant attention in the recent past, several weaknesses have been exposed in the blockchain fabric and exploited by malicious actors. Consequently, a number of efforts have been made to review and identify security threats for blockchain. For instance, Li et al. [11] carried out a thorough study on the major attacks on blockchain, highlighting loopholes in the proof of work based consensus system in terms of $> 50\%$ 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 breaches in the design of the transaction have been identified as a source of threats for blockchain-based applications. Minhaj [12] discussed various types of threats for blockchain from the perspective of different logical layers per varying processing and functioning zone of IoT devices. Authors classified the attacks in the low, intermediate, and high-level zones mapping the attacks into different layers of IoT security issues. Furthermore, Iuon-Chang [13] highlighted various security challenges of blockchain which can be used to carry out attacks. Authors specifically highlighted the risk of $> 50\%$ attack, forking problems, scalability of blockchain which can have a significant impact on blockchain security.

The efforts summarized above identify, collate and categorize potential threats for a typical blockchain system; however, to the best of our knowledge, there has not been any research which specifically discusses the impact of transaction malleability attack on different application domains. Consequently, in this paper, our focus is on the transaction malleability attacks for blockchain which can be considered as software design-based attack and can potentially lead to double-spending attacks. In particular, in transaction malleability attack, the ID of the transaction 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 [1]. The consequence of a successful transaction malleability attack can be in the form of double-spending. As illustrated by Rosenfeld [14], in a successful attempt to perform double spending attack, the receiver typically is made to believe that their transaction has been confirmed whilst attempting to get another similar transaction (malleable transaction) accepted by the network (we explain it further in Section 5). The merchant (beneficiary

of the transaction or the person responsible for receiving the asset through transaction) in this case will not receive the intended coins.

One of the important factors to note is that although the underlying technology for all the applications of blockchain is same, the impact of transaction malleability attack differs widely across these areas of implementation. An e-voting application is considered more sensitive as compared to cryptocurrency due to the sentiments of people and the role of voting in modern democracies. For instance, the interference in the 2016 election in the U.S. that resulted in the dismissal of 35 Russian diplomats shows the significance of the e-voting and the real-world consequences of disruption in the expected standard. As there is no fixed timeline for the arrival of new transactions in blockchain (due to factors such as network delays), double utilization of a vote may occur [15], which can be easily facilitated by transaction malleability attack as witnessed in the case of Mt. Gox Exchange [16]. Such attack may be planned by controlling block generation rate to create an inconsistent version of blockchain (through blockchain forking) and may further be supported through selfish mining [17] (feasible if the system has designated and suspected miners). Apart from this, even if the blockchain restores itself to its consistent state, the resources expended for honest mining(taken over by selfish mining during the attack) can be significant [18] and may also lead to slowing down the system, eventually losing voter participation. Another significant potential threat in blockchain-based e-voting is early verification of the voting transaction. This is because mostly a genesis transaction of assigning a voting asset to the voter is expected to be closer(in terms of several intermediate transactions) to the transaction moving voting asset from a voter to the candidate in contrast to Bitcoin or any other cryptocurrency-based transaction moving Bitcoin to pay to the receiver [5]. This early verification of transaction may support forking which can lead to double spending and wastage of computing resources. Therefore, transaction malleability attack has the potential to sabotage the democratic process of voting and requires dedicated efforts to mitigate against it effectively.

3. Related works

Double spending attack is the possibility to spend a transaction twice or more as a consequence of transaction malleability. One of these transactions will be included in the public ledger while others will be considered invalid and discarded by the network. One way to assess the impact of transaction malleability in the blockchain is to artificially inject multiple malleable transactions immediately after an original (honest) transaction by changing the nonfunctional fields i.e. fields that do not change the semantics of the transaction, such as *sender's address, receiver's address, 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 consensus bug issue regarding OpenSSL was indirectly fixed by BIP66 and has been active from block 363,724 which was added to the blockchain on July 4, 2015 [19]. 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 the substantially similar transaction. If any of the malleable transactions gets 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 [14].

Within this context, the solution proposed by Andrychowicz and Dziembowski [1] for handling transaction malleability requires elimination of important script of the transaction which may cause malleability. The script is removed before the computation of hash which is signed by the author and is, therefore, expected to be a significant cost to compromise. Decker and Wattenhofer [18] investigated the incident of Mt. Gox regarding transaction malleability attack 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. [20] presented a solution to counter the problem of transaction malleability in Bitcoin and proposed alterations in the process of computing a final transaction hash to identify a transaction. Specifically, they proposed calculating hash value separately without using *scriptSig* field of a transaction (to be appended to the hash of the transaction generated through existing method) as this field gives an attacker the liberty to add or replace keywords which do not change the semantics of the transaction. Although this solution proposes 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) [16] was initially proposed as a solution to address scalability challenge in Bitcoin; however, it is also considered a defence against transaction malleability attack due to signature malleability. This is because the organization of data for signature of transaction is separated, and not a part of the transaction which is called witness. Due to this, it is also considered as a useful barrier to execute a transaction malleability attack successfully. One of the major problems of using SegWit [16] is compatibility i.e. the transactions following this scheme remain separated from the conventional transactions of blockchain and has split up the Bitcoin community into two groups; Bitcoin and Bitcoin Cash (BCH). As BCH community does not use SegWit, transaction malleability is still a problem in Bitcoin cash. Although the recent attack by BitClub in 2017 highlighted the need for SegWit to avoid transaction malleability attack in Bitcoin, the challenge still exists for developers who believe that the attack is possible even after the implementation of SegWit [16].

Recently, many attempts have been witnessed to disrupt and tamper electoral processes. For instance, incidents such as alleged external interference in the U.S. 2016 elections resulted in President Obama's decision to dismiss 35 Russian diplomats [21]. Consequently, researchers have investigated use of technology to strengthen such democratic processes and protect against disruption to the desired operation leading to the development of major projects, such as My Vote [22] and BitCongress [23]. 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 [24] where reward is given by enforcing smart contracts over the distributed network but it affects the actual freedom of ballot. Recently, there has been several approaches in blockchain-based online voting which provide secrecy of the ballot including [25] and [26] which work on the secrecy of ballot. With respect to double utilization in e-voting, existing approaches rely on the assumptions that the majority of computing resources are controlled by honest miners. Therefore, to the best of our knowledge, we believe that the processing sub-domain of electronic voting is an area where further scientific contributions are required to specifically target the possibilities, and potential of double utilization of tokens in a blockchain-based decentralized network.

Double-spending attacks are unavoidable in blockchain as they are subject to the Fischer, Lynch, Paterson (FLP) impossibility result [15], which inferences that consensus cannot be made in distributed systems that do not provide a fixed timeline for new message arrival i.e. new blocks. Transaction malleability attack may serve as a platform for further malicious activities such as blockchain-forking and disbelieving the voter that his vote has not been caste. Therefore, our focus in this paper is to simulate transaction malleability attack to highlight the need for further research concerning protection mechanisms to mitigate against this threat.

4. Attack model for transaction malleability

Transaction malleability is fundamentally a blockchain-based attack agnostic of the application domain. Existing instances of transaction malleability attack have targeted Bitcoin, such as Mt. Gox exchange and BitClub in recent years. Therefore, although the experimentation conducted in this paper has been simulated with an e-voting application (specifically a public voting scenario), the outcomes are valid for wider application domains. Furthermore, we have studied existing efforts within blockchain security analysis in general and analysis of blockchain-based e-voting in particular to identify current practice. For instance, Khader [27] carried out a statistical evaluation of existing protocols and discovered that their solution was able to address the issue of voter entitlement and utilization of voting token more than once (a malleable transaction has a tendency for double utilization of vote). Furthermore, there have been several approaches in blockchain-based online voting which provide secrecy of the ballot, such as [28] and [29]. With respect to double utilization in e-voting, these approaches rely on the assumptions that the majority of computing resources are controlled by honest miners. Therefore, it is believed that further scientific advancements are needed within e-voting domain to specifically target the potential of double utilization in a blockchain-based decentralized network [30]. In this context, the experimentation presented here can help researchers investigate effective countermeasures to address this significant challenge.

The attack model for carrying out transaction malleability is 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*, got 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 under a realistic assumption. Fig. 1 shows the attack model for the given scenario.

Fig. 1 shows the scenario of our attack model which causes a malleable transaction to get into a blockchain block. Furthermore, Fig. 2 shows the process model diagram showing the possibility of transaction malleability attack in a controlled but realistic environment. In order to model the above mentioned scenario, we can make following assumptions:

- The system follows a strong secure cryptographic scheme, such as [25]. Transactions in the blockchain are identified by their hashes, which are generated by hashing the transaction data twice using the Secure Hash Algorithm (SHA-256), as explained in Section 5
- Hash rate refers to the computational strength of a machine. In this case, it has been kept consistent by sending honest and malleable transactions using the same machine to provide an equal opportunity in terms of hash rate for both types of transactions to get mined as explained further in Section 7.
- Difficulty level has been kept constant by pre-calculating the nonce value against the same pre-calculated hash. This had to be maintained constant (as illustrated in Fig. 15) along with hash rate to observe the impact of transaction mining time, which is the most critical factor for carrying out a successful attack as explained in Section 7.

The blockchain system has been designed for transaction databases (usually financial transactions, such as Bitcoin), which are publicly shared by all the nodes in the network. Each transaction contains a transaction ID, which is hash of all the fields in a transaction. It has been studied recently [1,6] that signed transactions are slightly *malleable* such that it is possible to modify a signed transaction in certain minor ways without invalidating the signature. Cryptography ensures that the critical details about a transaction, such as sender, receiver and amount cannot be changed but certain non-functional fields that do not contribute to the critical parts of a transaction may be changed, which cause the hash (transaction ID) to be changed

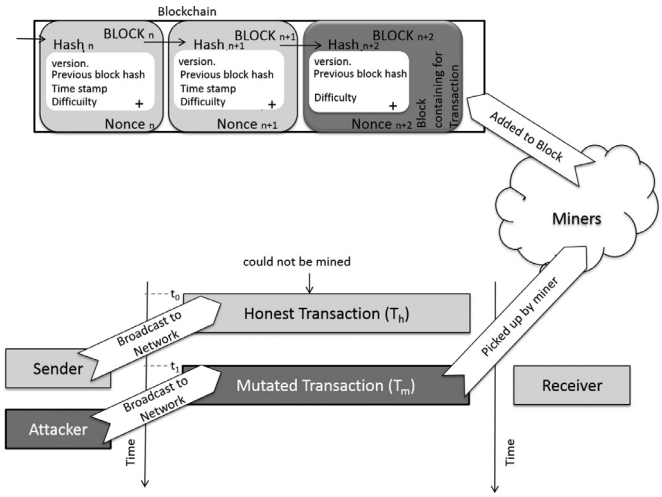

Fig. 1. Simulation of attack model for transaction malleability.

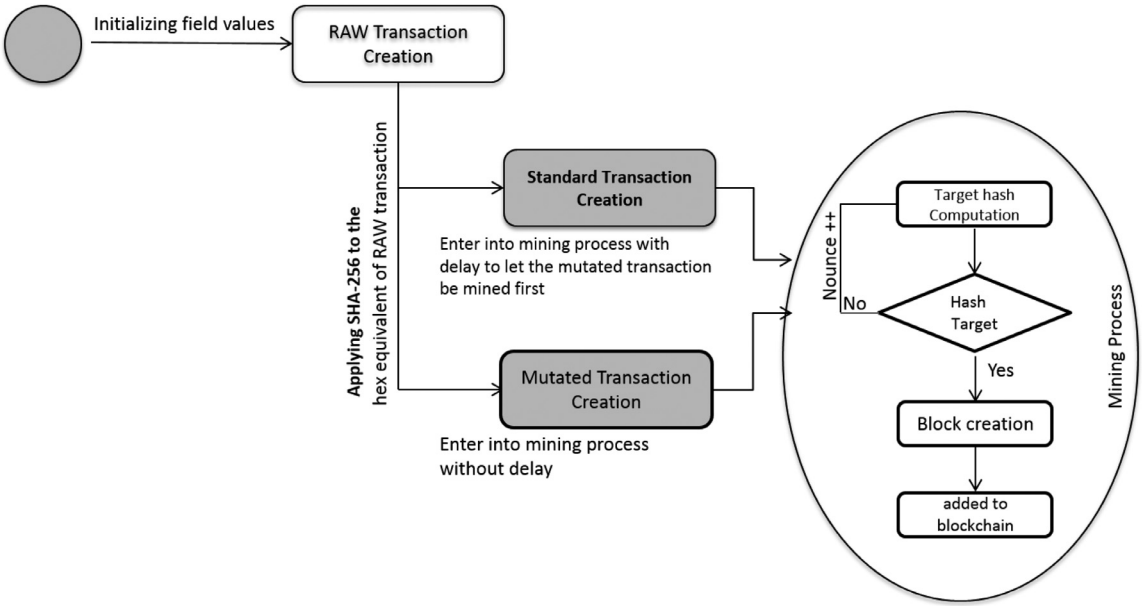

Fig. 2. The process model diagram for potential transaction malleability attack.

for the same transaction. This answer depends upon the domain of the application under focus and the platform upon which it is being run. For instance, in the case of Bitcoin, the transaction structure depends on the fields compatible with the exchange of Bitcoin. In this context, the fields which do not contribute in the amount being transferred between the sender's and receiver's addresses are treated as non-functional fields. Therefore, to avoid transaction malleability, one should not accept the transactions having fewer than an acceptable number of confirmations because all the following transactions in a blockchain depend upon the hashes of the previous transactions, and those hashes can be changed until they are confirmed in a block.

5. Simulation of transaction malleability attack

The success of the above defined attack model depends on the percentage in which the mutated transaction gets mined and the original transaction gets rejected by the miners on the network. This success is measured by the number of miners that receive the mutated transaction first and original transaction later. In order to achieve success, the mutated transaction needs to be broadcasted to the network much faster than the original transaction. In this section, we present detailed explanation of different aspects of the experimentation setup.

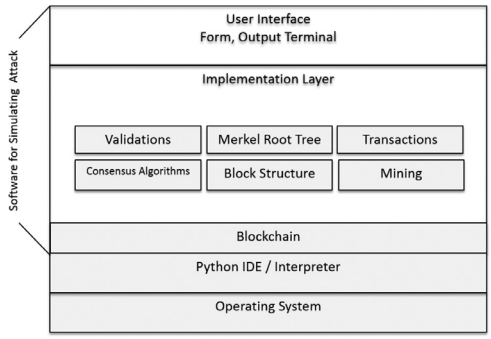

Fig. 3. System architecture for experimentation.

Table 1
Hardware and software specifications.

S. No.	Platform	Hardware Specification		
		Processor	Memory	Page File
01	Windows 10 Pro 64-bit	Intel Core i3-4005u CPU @ 1.70GHz (4CPUs)	4096MB RAM	5586MB Used 1887MB available
02	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

5.1. Conditions for successful transaction malleability

Transaction malleability attack is possible if the attacker manages to change transaction, let's say *Th*, to transaction *Tm*, which is syntactically different from transaction *Th* but semantically similar to it. This can be achieved by exploiting the malleability of signature schemes used in Bitcoin transaction. One of the transactions will be included in the public ledger whereas other will be discarded by the network as it will be considered invalid. In order to achieve a successful transaction malleability attack, the following scenarios are possible.

- One way to assess the feasibility of malleability impact in Bitcoin is to artificially inject multiple malleable transactions immediately after an original transaction by changing the non-functional fields, such as Sender's address, Receiver's address and amount. These are all cryptographically signed by the sender but can result in modification of the hash of the TxID. Now, if any one of the malleable transactions gets mined prior to the original transaction, the miners will add this transaction to the block as a valid one because the critical fields in the transaction were unchanged. The hash value changes when non-functional fields change due to collision resistance property. A typical Bitcoin transaction is in the form of (message M, signature S on M). Attackers compute another valid signature let's say S' for transaction *Th*, by doing so another *Tm* is created from (message M, signature S'), which is a valid transaction with the same message as *Th* but with different signature.
- Another way of creating mutated transaction is by adding dummy PUSH and POP instructions to the signature S producing S', which is operationally equivalent to S but from the syntactic point of view it is different. The attacker, after creating mutated transaction, broadcasts it to the network. If the attacker is lucky then this mutated block will be confirmed first and miners will include *Tm* into the blockchain instead of *Th*.
- Transaction malleability attack model can also be successful in blockchain if the sender uses hash or transaction ID to track their transaction in the public ledger and resend the assets, if the expected transaction ID is not found in the blockchain public ledger.

5.2. Experimentation to simulate TM attack

In order to simulate successful transaction malleability attack, we have used libraries available within Python following the major mathematical processes necessary for simulating basic characteristics of a typical blockchain network, such as creation of genesis transaction, creation of genesis block hash, creation of honest and mutated transaction and their cryptographic schemes. The overall architecture of the system used for this experimentation is presented in Fig. 3 whereas details of the hardware and software used are presented in Table 1. With respect to experimentation with blockchain in a real-life scenario, hardware specifications affect the hash rate in proof of work based blockchain systems. However, for these experiments, the setup has been controlled by sending honest and malleable transaction from the same machine to observe the impact of transaction mining due to delayed arrival of transaction. Therefore, with respect to this experiment, any change in the hardware specification for honest or malleable transaction is not expected to cause any significant change in the results.

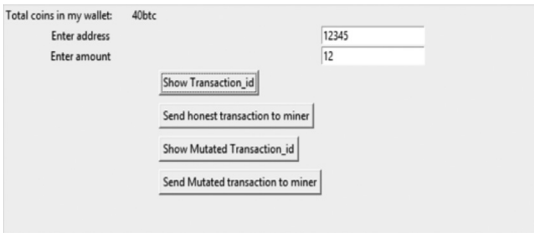

Fig. 4. Sample application interface.

Attributes	Values—Taken
Versionfield	01000000
Noofinputs	01
Previoustransactiono utputhash	be66e10da854e7aea9338c1f91cd489768d1d6d 7189f586d7a3613f2a24d5396
Nextsequence	00000000
Lengthofscriptsig	6b
Scriptsig	76a914dd6cce9f255a8cc17bda8ba0373df8e861 cb866e88ac
Sequence	Ffffffff
Noofoutputs	01
Amount	Entered by the user , here it is 12
Lengthofoutputs	19
Outputscript	Address for transfer here it is ; 12345
Locktime	00000000
Hashcodetype	01000000

Fig. 5. Attributes/ values for raw transaction.

Factors such as background processes, memory utilization, running services on the same machine are not expected to vary significantly while sending honest and malleable transaction.

5.2.1. General assumptions for TM attack

In order to develop the above mentioned scenario, the system has been tested and run on a single machine so that the hash rate, mining process and difficulty level (adjusted through pre-calculated real values) can be kept the same. The delay is induced programmatically (to simulate network latency) to enable the mutated transaction mine earlier than honest transaction.

In this simulation, we have used the Proof of Work (PoW) as consensus algorithm which is used in Bitcoin blockchain for its mining process. A proof of work includes finding data which consumes energy, cost, time and takes significant effort to compute but is very easy to verify. A large number of trial and error iterations are executed to eventually find the acceptable hash of the new block. Difficulty should be adjusted at a particular level to provide equal ground for honest and mutated transaction to be mined, and to limit the rate of generating new blocks.

Difficulty is stored in blocks in packed representation and every miner must decode into hexadecimal to find target hash for that specific block. We are considering 0x19015f53 as bits to set a difficulty level where we already have the knowledge that an acceptable hash will be generated at a certain value of nonce. When the miners look to start mining, they actually enter into a race of creating a new block header hash so that they may be able to get their block entered into the blockchain. Miners start building the header using the Merkle root hash (history of all the transactions) of 32 bytes in length, version number which is of four bytes in length, hash of last block in the chain which is also of 32 bytes in length, Timestamp (a number showing total time elapsed since 1970-01-01 00:00) and the nonce, a four byte random number which is generated and repeated to find the required hash.

5.2.2. Producing honest transaction

Generating raw transaction: The process to produce honest transaction starts by seeking user input for asset and the address. The graphical interface of our application is presented in Fig. 4. Here, the asset is assumed to be an amount whereas the address will be used later on to initialize the output script parameter.

In order to achieve our objective to conduct simulation process of honest and controlled mining, certain realistic values of several parameters have been assumed which contain same number of bytes (as it is used in the context of Bitcoin) so that a raw transaction (distributed to the miner) may be created. This raw transaction will be later used to create transaction ID by applying SHA256 of encoded hexadecimal digest available in *hashlib* class of Python. We produce a raw transaction by concatenating version field, number of inputs, previous transaction output hash, next sequence, length of script signature, script signature, sequence, no of outputs, amount(transferred), length of outputs, output script, lock time, hash code type. Fig. 5 presents the attribute/value pairs used to produce the raw transaction whereas Fig. 6 presents the raw transaction generated as a result of this experiment.

```
0100000001be66e10da854e7aea9338c1f91cd489768d1d6d7189f586
d7a3613f2a24d5396000000006b76a914dd6cce9f255a8cc17bda8ba0
373df8e861cb866e88acffffffff01121912345000000000001000000
```

Fig. 6. Raw transaction hash.

```
3aee278c8e8f68c9222cc5efff399478b6c1d14386af7fd3
ae04a7b72b27f243
```

Fig. 7. 64-bit transaction ID.

Attributes	Values--Taken
Versionfield	01000000
Noofinputs	01
previoustransactio noutputhash	be66e10da854e7aea9338c1f91cd489768d1d6d7189f586d7a 3613f2a24d5396
Nextsequence	00000000
Lengthofscriptsig	6b
Scriptsig	76a914dd6cce9f255a8cc17bda8ba0373df8e861cb866e88ac
Sequence	ffffffff
Noofoutputs	01
Amount	Entered by the user , here it is 12
Lengthofoutputs	**19 + "00"**
Outputscript	Address for transfer here it is ; 12345
Locktime	00000000
Hashcodetype	01000000

Fig. 8. Attributes/Values for malleable transaction.

```
Raw Tx:   0100000001be66e10da854e7aea9338c1f91cd489768d1d6d7189f586d7a3613
          f2a24d5396000000006b76a914dd6cce9f255a8cc17bda8ba0373df8e861cb86
          6e88acffffffff01121900123450000000001000000

64-bit    45dd6a53129e51f2fca8538819d6f45449fa597d3bf1429
Tx ID:    3d30fb1e68fa8e300
```

Fig. 9. Malleable raw transaction and 64-bit ID.

Generating a 64 bit Transaction ID: In order to get transaction ID from raw transaction, we applied SHA256 of encoded hexadecimal digest available in *hashlib* class of python and display it on output. The 64-bit transaction ID generated through this process is presented in Fig. 7.

This represents an *honest transaction* and is used to later create another transaction which will be semantically same but will produce a different hash as its transaction ID. We will call that transaction as a *mutated or malleable transaction*. The race will then start (although depending upon several factors which will be discussed later, such as network latency, picking up of transaction by miner to solve proof of work) regarding which transaction gets mined and become a part of blockchain in the new block.

5.2.3. Producing malleable transaction

Generating raw transaction: As we discussed in the previous sections, a transaction can be made malleable by changing it in a way that it remains semantically same but produces different hash as its transaction ID. In this experiment, we produced malleable transaction by appending 00 in the creation of malleable raw transaction for the original honest transaction we generated above. Therefore, in this experiment, transferring amount and address fields, which are being taken as input are same i.e. address is 12345 and transferring amount is 12. The only difference in producing the raw transaction of honest and malleable transaction is the appending of 00 just after the parameter of length of outputs as shown in Fig. 8 while the rest of the parameters are same as those were used earlier for generating honest raw transaction. The raw transaction that we get here in this experiment is presented in Fig. 9.

Generating 64 bit transaction ID: The encoding procedure was replicated from generating raw transaction by applying SHA256 of encoded hexadecimal digest available in *hashlib* class of Python. The raw transaction and the 64-bit transaction ID generated as a result is presented in Fig. 9.

5.2.4. Target hash computation for mining

Mining is effectively a process of calculating the hash of the newly proposed block by the miner, which is less than the target hash. Since header creation has most of its fields fixed, we can assume some realistic values of certain parameters that will be used for block headers creation. The fields which we have used for header's creation include version, previous block hash, Merkle root tree hash, timestamp, bits for setting up difficulty level and nonce. Fig. 10 shows the values, which have been assumed for creating new block header.

Attributes	Values--Taken
previous block hash	000000000000000117c80378b8da0e33559b5997f2ad55e2f7d18ec1975b9717
merkle root tree hash	871714dcbae6c8193a2bb9b2a69fe1c0440399f38d94b3a0f1b447275a29978a
time stamp	0x53058b35
bits	0x19015f53
nonce	Arbitrary Number which is incremented by one in each iteration

Fig. 10. Attributes/ Values for creating block header hash.

Fig. 11. Attributes & Values for creating block header hash.

The block header hash in this case contains only one variable, which will be iterated to get a hash less than the current target. We will now compute the current target by calculating difficulty level in the context of how it is calculated for Bitcoin implementation. Firstly, a miner uses the packed illustration of block to get the current target as hash by deciphering this packed illustration. In our experiment, we have considered a known and familiar value of this packed illustration of block i.e. 0x19015f53' for simplicity. Separating first two numbers from this packed version, we get 19 (in hexadecimal) and if we convert it into decimal we get 25. The remaining number in this packed format is 015f53' and if we convert it into base 10 we get 89939. After working out these values by bitwise right shift(24 shifts), performing AND operation (with 0xffffff), putting it in a difficulty formula and getting it in a 64 bit hexadecimal format we get current target as: 0000000000000015*f*5300000000000000000

00000000000000000000000000.

The computed hash of newly proposed block header must be less than this target. Our mining process will start computing this block header hash for various incrementing values of nonce in a quest to generate required hash. In order to enable the malleable transaction to get mined first, an artificial delay was induced just after transaction formation but before the start of the mining process of this transaction to mitigate as a network delay. To make use of this delay, another instance was used (within the span of the delay) to send the malleable transaction with an immediate start of its mining process. In this way, in a simulated but with a realistically assumed environment, a malleable transaction was able to become part of the blockchain.

Since the value of nonce required for the desired hash was already known and the program instances were run on the same machines, the only significant factor was the time when the transaction was picked up by the miner. In this case, a race started between an honest transaction(with an artificially injected delay) and a malleable transaction (without any delay) by keeping the difficulty level and hash rate same in both the runs. In this way, the transaction malleability attack was successfully simulated. Fig. 11 displays the final output of the successful transaction malleability attack when nonce reaches to a value of *856192328*.

5.2.5. Malleable transaction mining

As explained in Section 4, we assume that the malleable transaction was sent after honest transaction but was mined before original (honest) transaction. This shows that although the honest transaction existed before sending malleable transaction but was mined after the malleable transaction. In real world, it may happen due to;

- Possible network latency, which may cause malleable transaction to reach earlier to the miner.
- Due to early picking of a malleable transaction from the mining pool by the miner. The addition of new blocks (event) follows Poisson distribution and is completely independent of each other.
- Attacker gets lucky to mine his transaction into the block. For instance, possibility that the miner who picked up the malleable transaction finds the required nonce earlier than the miner who picked up honest transaction.

For the purpose of experiments presented here, the above real world possibilities have been modelled in the following way. A race was started between an honest transaction (with an artificially injected delay) and a malleable transaction

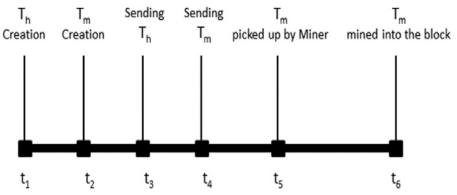

Fig. 12. Example Timeline for successful transaction malleability attack.

```
856192320 60c86d7767a9280b26bdc65a95957b4e6bf130731651ad42195b5f567e27ee6d
856192321 5c962a229ef13af8861f4694f5a2f58c3ab69139861aa672db95d9fa0bd592c3
856192322 94c3a385db717b75e2a81f4ba5819920fbc360ccfa96e265d27a46d5e0c9dc9a
856192323 cace4fce6138921f4a368d254600ff756077b3eb8dbfe178a4c3e6ec3427b86c
856192324 d2780b4afff9c2bdf733f0889843cbc608fd119a357605bef1572a11adbf12a1
856192325 b55b0b4564312bfe889780bfc408597d36950badfe8eeac5060289c36e0b0f0b
856192326 c03df18fca119ea22778c65f6116a05416e4d17acbaf516a1bf33c78be79140d
856192327 8cca5b98c8adeb059907c6b94dd513e72c51d0fd2f95922b57004f567d10e497
856192328 000000000000000e067a478024addfecdc93628978aa52d91fabd4292982a50
successful Honest Transaction
ff33ff4af2f455fd285c47972efd042b2d818f672efa21604a83d37d6639ec24-->000000000000000e067a478024addfecdc93628978aa52d91fabd4292982a50
Elapsed time 0:00:00.102000
```

Fig. 13. Sample experiments to demonstrate elapsed time for honest transaction.

(without any delay) by keeping the difficulty level and hash rate same. This is equivalent to the real world condition of network latency and therefore enables our setup to simulate close to real-world scenario.

Another important factor which was helpful for successfully carrying out transaction malleability attack was to win the race for solving mathematical puzzle (proof of work) in case of malleable transaction (by the miner), as mentioned above. This was achieved by making less number of iterations to find the correct nonce for malleable transaction in contrast to have more iteration for honest transactions (through pre-calculation of required nonce). Fig. 12 represents the time series for the occurrence of events responsible for carrying out a successful transaction malleability attack. It can be seen in Fig. 12 that the malleable transaction (which was created and initiated after the honest transaction was sent to the blockchain), won the race and was picked up by the miner before its original version and got mined into the block. The success of attack was achieved by simulating network delays for honest transaction, and by making miner of malleable transaction lucky in calculating solution of mathematical puzzle for malleable transaction as it happens in real world (by reducing number of iterations for finding the required target hash).

6. Experimentation results

The experiment was conducted while observing several performance indicators, which may play a key role in running the internal background processes behind blockchain. The operational characteristics of blockchain in this experiment were recorded in the following way.

In order to observe the behavior of the blockchain in terms of the time elapsed for producing new blockhash to mine the honest transactions, different numbers of independent iterations were performed to draw a pattern and a comparison with the same pattern for malleable transactions. The Fig. 13 below shows how the data for elapsed time was recorded individually for each honest transaction at different time instances when the desired hash of the block is found. It can be seen from this figure that the elapsed time ranges from 71 ms to 102 ms against an honest transaction when it was run on the same machine in each trial.

Another important aspect to notice is that the nonce value for the required blockhash is already known in this simulation so that the computation may be performed at a reasonable time. The required number of iterations to reach this nonce value was fixed across all the attempts for finding out the elapsed time for all the honest transactions as the starting nonce value was fixed to achieve the value lower than the target hash at a realistic time. Therefore, the computational power in terms of hashes per second of the machine (which is running simulation) could be found against every mined transaction. In every attempt of finding out the elapsed time, the conditions were almost similar except the independent parameters of machine which are related to the state of machine which may vary and can affect the performance of the system like background processes, memory utilization and running services. A number of experiments were conducted and a sample is presented in Fig. 13 which demonstrates behavior of the system when the honest transactions took more time to mine than their respective malleable transactions. Similarly sample experiments presented in Fig. 14 show the behavior of the system when the malleable transactions were being processed.

These sample measurements were taken to facilitate the mining of malleable transactions earlier not by any artificial programming delay in case of honest transactions but by setting up the starting nonce value at a number, which is larger than the number was used to mine the respective malleable transaction. This is much better approach to discourage honest and encourage malleable transaction while simulating and recording data for a shorter elapsed time in case of malleable transactions than the honest transactions. This was done with a realistic assumption that the attacker node would likely reach the desired block hash in a quicker period of time than the victim's node so that the chances of a successful transaction malleability attack may be increased.

```
856192320 60c86d7767a9280b26bdc65a95957b4e6bf130731651ad42195b5f567e27ee6d
856192321 5c962a229ef13af8861f4694f5a2f58c3ab69139861aa672db95d9fa0bd592c3
856192322 94c3a385db717b75e2a81f4ba5819920fbc360ccfa96e265d27a46d5e0c9dc9a
856192323 cace4fce6138921f4a368d254600ff756077b3eb8dbfe178a4c3e6ec3427b86c
856192324 d2780b4afff9c2bdf733f0889843cbc608fd119a357605bef1572a11adbf12a1
856192325 b55b0b4564312bfe889780bfc408597d36950badfe8eeac5060289c36e0b0f0b
856192326 c03df18fca119ea22778c65f6116a05416e4d17acbaf516a1bf33c78be79140d
856192327 8cca5b98c8adeb059907c6b94dd513e72c51d0fd2f95922b57004f567d10e497
856192328 0000000000000000e067a478024addfecdc93628978aa52d91fabd4292982a50
success for Malleable Transaction
ff33ff4af2f455fd285c47972efd042b2d818f672efa21604a83d37d6639ec24-->0000000000000000e067a478024addfecdc93628978aa52d91fabd4292982a50
 Elapsed time 0:00:00.004000
```

Fig. 14. Sample experiments to demonstrate elapsed time for malleable transaction.

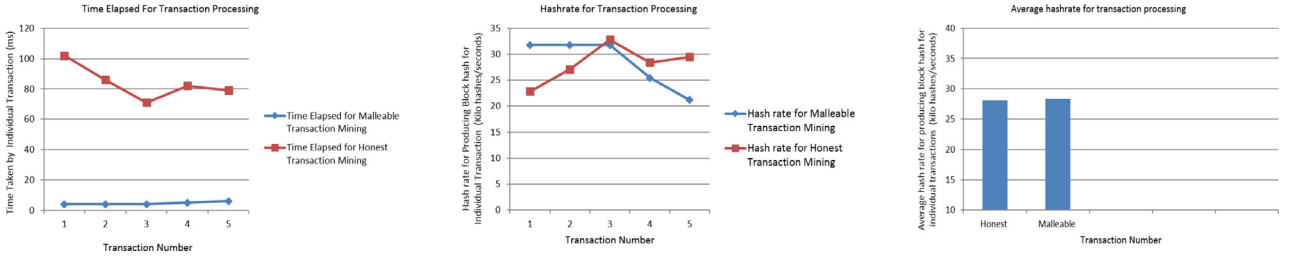

Fig. 15. Sample experiments to demonstrate elapsed time for malleable transaction.

Table 2
Hashrate/mining time for honest transaction.

Tx. No.	Elapsed Time (ms)	Hash rate
1	102	22.8235
2	86	27.0697
3	71	32.78873
4	82	28.39024
5	79	29.6835

Table 3
Hashrate/mining time for malleable transaction.

Tx. No.	Elapsed Time (ms)	Hash rate
1	4	31.750
2	4	31.750
3	4	31.750
4	5	25.400
5	6	21.166

Therefore, at the time of execution of a successful transaction malleability attack, the support will not only come in the form of programming delay while releasing an honest transaction (to simulate network delay) but also in terms of added computational strength in the mining process of generating required block hash for a malleable transaction as the starting nonce value would have been shorter to reach to the solution quickly (for simulating more hash power). In case of honest transaction, it was programmed to run 2238 iterations find the solution while in case of malleable transactions the required iterations to find the required block hash were programmed to be 127 (since the final nonce value was pre-calculated). The iterations reflect to the number of required hashes unless the final hash is found.

Tables 2 and 3 show the tabulated data for individual honest and malleable transactions along with their elapsed time and hash rate (number of hashes per second) respectively. Hash rate was calculated by dividing the number of generated hashes to the total elapsed time until the required blockhash is produced in order to get the number of hashes per unit time (in seconds). The data was recorded without any programming delay and the elapsed time for malleable transactions was reduced due to lesser number of required hashes than the respective honest transaction by manipulating nonce values for both of these cases.

7. Discussion and analysis

As discussed in Section 6 the elapsed time was programmatically reduced to assess the feasibility of transaction malleability. Since the starting nonce value for honest transaction was not much closer to the final nonce value for producing required acceptable hash in contrast to the case of malleable transaction, the elapsed time was decreased for malleable

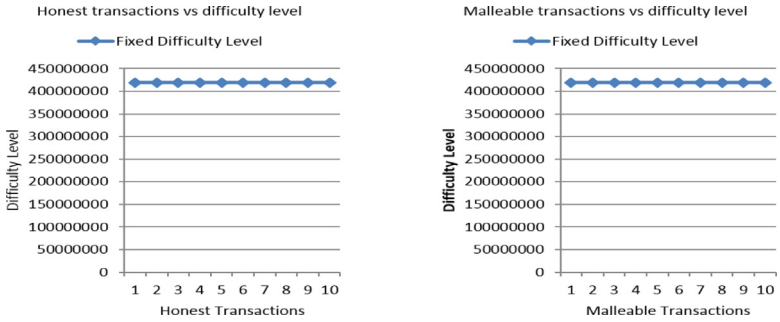

Fig. 16. Constant hashrate for honest and malleable transactions respectively.

transactions. This factor can reasonably increase or decrease the chances for the success of the transaction malleability attack.

In a typical attack scenario, the attacker's machine is expected to calculate the desired block hash earlier than the honest machine i.e. it should take lesser time to compute the required hash. In this simulation, the scenario is managed by requiring the attacking instance of the program to compute fewer steps than the honest instance. Consequently, the graph in Fig. 15. A shows the difference between the elapsed time for honest and malleable transaction. It can be seen in the graph that the maximum value of elapsed time for a malleable transaction is 4 ms, which is much lower than the minimum value of elapsed time for an honest transaction i.e. 71 ms. It means that in this simulation run, the malleable transaction needs to be mined at least approximately 18 times (17.75 times) quicker than the existing elapsed time if it needs to rationally increase chances of denying a transaction malleability attack. Similarly, another very interesting inference may be made from this statistic that rather than expecting an attacker to slow down his hash rate for a failed attack (which is highly unlikely), an effort may be made to increase the hash rate at least to a level to outwit the attacker. This implies that in this case if the elapsed time for mining the honest transaction is reduced to a percentage of 97.183% of the current minimum operational elapsed time which makes almost 69 ms and would make elapsed time closer to 4 ms, then there is a chance for the honest machine to put a competition to the attacker otherwise the current condition supports heavily to a successful transaction malleability attack.

As it has been stated earlier, in the current simulation, the difference of elapsed time has been made by starting with a close nonce value for malleable and a relatively larger value of honest transaction so that the honest instance of the program would have computed more hashes than its equivalent dishonest instance for the malleable version of the transaction. In a real world scenario, this can be achieved by migrating honest node to a more powerful machine of a higher hash rate. Since in this case of simulation, almost 97% improvement is required to challenge the attackers instance of malleable transaction. An important aspect to note here is that the level of challenge was not same for the honest and malleable instances in terms of computational work (required number of hashes to propose the block). This was required to simulate the successful attack along with programmatically induced delay for honest transaction. Nevertheless the difficulty level and hash rate (run on same machine) were kept same to observe the behaviour of the system in the context of delay and elapsed time for transaction processing.

Fig. 15.B and .C shows the hash rate against for the individual and average transactions. The average hash rate was 28.108 kilo hashes/second and 28.363 kilo hashes/second for honest and malleable transactions respectively which is almost similar as the simulation was run on the same machine. The hash rate and difficulty level had to be maintained constant across honest and malleable running instances so (as illustrated in Fig. 16 that the comparison and inferences may be based upon the transaction mining time in terms of controlled nonce values (for indirectly controlling the computational steps on the similar machine platform) and then finally inducing some delays programmatically in honest transaction release to ensure a successful transaction malleability attack.

8. Conclusion

Blockchain technology has attracted significant interest from researchers across diverse domains to address challenges fundamentally concerning trust, non-repudiation and auditing. However, transaction malleability has emerged as a threat to blockchain that can corrupt the blockchain state resulting in attacks, such as double-spending. This paper has focused on investigating the feasibility of transaction malleability attack in blockchain, aiming to identify scenarios that may lead to a successful transaction malleability attack. Through our efforts, we have highlighted conditions causing such attack to facilitate the development of protection mechanisms for them. Specifically, this paper has presented transaction malleability attack within the context of a chosen application domain i.e. blockchain-based e-voting. We implemented the sample application use-case as well as blockchain simulations using Python libraries. The evaluation identified the role of parameters, such as network delay and block generation rate in achieving a successful transaction malleability attack. It, therefore, showed that the transaction malleability attack can produce an inconsistent blockchain, encouraging attackers to exploit

the situation for a variety of malicious purposes. We are continuing our research with further evaluation using a real-life test-bed for the approach presented in this paper.

Declaration of Competing Interest

None.

References

[1] Andrychowicz M, Dziembowski S, Malinowski D, Mazurek Ł. On the malleability of bitcoin transactions. In: Brenner M, Christin N, Johnson B, Rohloff K, editors. Financial Cryptography and Data Security. Berlin, Heidelberg: Springer Berlin Heidelberg; 2015. p. 1–18. ISBN 978-3-662-48051-9.
[2] Suliman A. Monetization of iot data using smart contracts. IET Networks 2019;8(5):32–7. https://digital-library.theiet.org/content/journals/10.1049/iet-net.2018.5026.
[3] Salah K, Rehman MHU, Nizamuddin N, Al-Fuqaha A. Blockchain for ai: review and open research challenges. IEEE Access 2019;7:10127–49. doi:10.1109/ACCESS.2018.2890507.
[4] Nizamuddin N, Salah K, Azad MA, Arshad J, Rehman M. Decentralized document version control using ethereum blockchain and ipfs. Comput Electr Eng 2019;76:183–97. doi:10.1016/j.compeleceng.2019.03.014.
[5] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. Cryptography Mailing list at 2009 https://metzdowdcom.
[6] Rosenfeld M. Analysis of bitcoin pooled mining reward systems 2011 arXiv:11124980.
[7] Chaum D. Secret-ballot receipts: True voter-verifiable elections. IEEE Secur Priv 2004;2(1):38–47.
[8] Hao F, A RPY, Zielinski P. Anonymous voting by two-round public discussion. IET Inf Secur 2010;4(2):62–7.
[9] Khan KM, Arshad J, Khan MM. Secure digital voting system based on blockchain technology. Int J Electron Gov Res 2018;14(1):53–62. doi:10.4018/IJEGR.2018010103.
[10] Shahzad B, Crowcroft J. Trustworthy electronic voting using adjusted blockchain technology. IEEE Access 2019;7:24477–88. doi:10.1109/ACCESS.2019.2895670.
[11] Li X., Jiang P., Chen T., Luo X., Wen Q.. A survey on the security of blockchain systems. Future Gener Comput Syst https://bit.ly/2q14WLW.
[12] Khan MA, Salah K. Iot security: review, blockchain solutions, and open challenges. Future Gener Comput Syst 2018;82:395–411.
[13] Lin I-C, Liao T-C. A survey of blockchain security issues and challenges. I J Netw Secur 2017;19:653–9.
[14] Rosenfeld M. Analysis of hashrate-based double spending. CoRR 2014. http://arxiv.org/abs/1402.2009.
[15] Fischer MJ, Lynch NA, Paterson MS. Impossibility of distributed consensus with one faulty process.. Tech. Rep.. Massachusetts Inst of Tech Cambridge Lab for Computer Science; 1982.
[16] Schoenfeld H.. Malleability attack and why it matters. https://bit.ly/348Rbti.
[17] Gobel J, Keeler H, Krzesinski A, Taylor P. Bitcoin blockchaindynamics: the selfish-mine strategy in the presence of propagation delay. Perform Eval 2016;104:23–41. doi:10.1016/j.peva.2016.07.001.
[18] Decker C, Wattenhofer R. Bitcoin transaction malleability and mtgox. In: European Symposium on Research in Computer Security. Springer; 2014. p. 313–26.
[19] Wuille P.. Disclosure: consensus bug indirectly solved by bip66. 2015. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2015-July/009697.html.
[20] Rajput U, Abbas F, Oh H. A solution towards eliminating transaction malleability in bitcoin. JIPS 2018;14:837–50.
[21] BBC. Russia's lavrov threatens us over seized diplomatic mansions. 2017. https://www.bbc.com/news/world-europe-40567687.
[22] Hourt N.. Blockchain technology in online voting. 2017. https://followmyvote.com/online-voting-technology/blockchain-technology/.
[23] Rockwell M.. Bitcongress process for blockchain voting & law. 2017.
[24] Zhao Z, Chan T-HH. How to vote privately using bitcoin. In: International Conference on Information and Communications Security. Springer; 2015. p. 82–96.
[25] Miers I, Garman C, Green M, Rubin AD. Zerocoin: anonymous distributed e-cash from bitcoin. In: 2013 IEEE Symposium on Security and Privacy. IEEE; 2013. p. 397–411.
[26] Ibrahim MH. Securecoin: a robust secure and efficient protocol for anonymous bitcoin ecosystem.. IJ Netw Secur 2017;19(2):295–312.
[27] Khader D, Smyth B, Ryan P, Hao F. A fair and robust voting system by broadcast. Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI) 2012;205.
[28] Hao F, Kreeger MN, Randell B, Clarke D, Shahandashti SF, Lee PHJ. Every vote counts: ensuring integrity in large-scale electronic voting. 2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 14). USENIX Association; 2014.
[29] Kiayias A, Yung M. Self-tallying elections and perfect ballot secrecy. In: Naccache D, Paillier P, editors. Public Key Cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg; 2002. p. 141–58. ISBN 978-3-540-45664-3.
[30] Agarwal N. Digital voting with the use of blockchain technology. https://www.linkedin.com/pulse/digital-voting-use-blockchain-technology-naman-agarwal/.

Kashif Mehboob Khangraduated in Computer Engineering from Sir Syed University of Engineering & Technology in 2005-2006 followed by Master in C.S. & I.T. from N.E.D University of Engineering & Technology in 2009. He is currently a Ph.D. student in information security at the N.E.D. University, Pakistan.

Junaid Arshad is an Associate Professor at the Birmingham City University, UK. He achieved his Ph.D. from the University of Leeds, UK in 2011. Junaid's research areas include distributed computing, high performance computing (grid and cloud computing) and Internet of Things emphasizing security challenges, including intrusion detection and response, trust establishment and management, and security event classification.

Muhammad Mubashir Khan Muhammad Mubashir Khan is an Associate Professor in the Department of Computer Science and Information Technology at NED University of Engineering and Technology, Karachi Pakistan. He received his Ph.D. degree in Computing from University of Leeds, UK in 2011. His current research interests include Network and Information Security, Cybersecurity and Quantum Cryptography.