J Inf Process Syst, Vol.14, No.4, pp.837~850, August 2018
https://doi.org/10.3745/JIPS.03.0101

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Solution towards Eliminating Transaction Malleability in Bitcoin

Ubaidullah Rajput*, Fizza Abbas*, and Heekuck Oh**

Abstract

Bitcoin is a decentralized crypto-currency, which is based on the peer-to-peer network, and was introduced by Satoshi Nakamoto in 2008. Bitcoin transactions are written by using a scripting language. The hash value of a transaction's script is used to identify the transaction over the network. In February 2014, a Bitcoin exchange company, Mt. Gox, claimed that they had lost hundreds of millions US dollars worth of Bitcoins in an attack known as transaction malleability. Although known about since 2011, this was the first known attack that resulted in a company loosing multi-millions of US dollars in Bitcoins. Our reason for writing this paper is to understand Bitcoin transaction malleability and to propose an efficient solution. Our solution is a softfork (i.e., it can be gradually implemented). Towards the end of the paper we present a detailed analysis of our scheme with respect to various transaction malleability-based attack scenarios to show that our simple solution can prevent future incidents involving transaction malleability from occurring. We compare our scheme with existing approaches and present an analysis regarding the computational cost and storage requirements of our proposed solution, which shows the feasibility of our proposed scheme.

Keywords

Crypto-currency, Bitcoin, Transaction Malleability

1. Introduction

After its introduction in 2008 by Satoshi Nakamoto, the Bitcoin crypto-currency has gained significant popularity amongst other various crypto-currencies. It is a decentralized currency that utilizes cryptography to validate itself [1]. Bitcoin is a peer-to-peer network. A Bitcoin transaction is a signed piece of data written in a scripting language that governs the transfer of Bitcoins from one party to another. After a transaction is created, it is relayed to the network and collected by verifiers, who are known as "miners." The job of the miners is to verify a transaction. They collect a number of transactions that are broadcast over the network over a period of time and put them together in a block. Upon verification, the block containing these transactions is added to an online public ledger known as Blockchain. If the transaction is not verified it is rejected by the miners [2,3]. Blockchain is a publically available ledger that consists of all the transactions ever made in the Bitcoin crypto-currency system. Blockchain is made of blocks that are verified by the miners. This mechanism is the basis for preventing

Manuscript received September 15, 2015; accepted January 19, 2016.
Corresponding Author: Heekuck Oh (hkoh@hanyang.ac.kr)
* Dept. of Computer Systems Engineering, Quaid-e-Awam UEST, Nawabshah, Pakistan ({ubaidullah, fizza_alvi}@quest.edu.pk)
** Dept. of Computer Science and Engineering, Hanyang University (ERICA campus), Ansan, Korea (hkoh@hanyang.ac.kr)

double spending in Bitcoins.

Moreover, no government or centralized authority governs Bitcoin. Instead, it is a completely distributed network. One of the major questions in the world of crypto-currencies is: "Who creates the money at first?" In Bitcoin, the miners, who verify the Bitcoin transactions, get the newly generated Bitcoins. These Bitcoins serve as a reward for their efforts they put in by spending computing resources during block verification. Currently this reward is set to 25 Bitcoins [4, 5]. Every 4 years this reward is cut in half. This procedure of cutting rewards in half will continue until the total Bitcoins issued reaches 21 million. At that point, the generation of new Bitcoins will be capped and miners will only get transaction fees for their verification efforts.

A Bitcoin exchange is a company that deals in the currency exchange of Bitcoins with other currencies, like the US dollar. One such company, Mt. Gox, claimed in February 2014 that they were hit by a bug known to exist in Bitcoin called transaction malleability [6,7]. Mt. Gox claimed to lose hundreds of millions of dollars worth of Bitcoins due to this transaction malleability-based attack. However, later they claimed to recover some of them. A similar incident occurred at another Bitcoin-based online market place known as Silk Road 2. The online black market was famous for its dealing in prohibited items, such as drugs, etc. The owners of Silk Road 2 also claimed to have lost their Bitcoins [7,8]. The victims claimed that this was due to the transaction malleability bug in the Bitcoin system itself and put the blame on the design of the crypto-currency. The Bitcoin developers argued that this bug was known since 2011 and that the software at Mt. Gox was not developed properly to handle such attacks [9].

At the time of this writing, 361,290 blocks have been generated, verified, and added to Blockchain. A hardfork [10] (i.e., a major change that makes previously processed blocks invalid) might not be welcome. Transaction malleability, on the other hand, is a weakness of the system and a solution to this problem could eliminate future attacks from exploiting this weakness. The contributions of this paper are given below.

- First, we thoroughly investigate transaction malleability (i.e., how such an attack is possible) by providing a complete understanding of Bitcoin transactions.
- Second, we propose a solution that is a softfork change [11].
- We also provide the proof of concept that our softfork solution eliminates transaction malleability without any (or little) additional cost.

This work is ongoing research on this issue. Our earlier effort was accepted in [12]. In this paper we are extending [12] by digging more deeply into the transaction malleability issue and we analyze our solution in more detail to show that it is practical enough to be used.

The rest of the paper is organized as follows: Section 2 outlines the related work and Section 3 outlines the Bitcoin systems, the concept of transactions, and Blockchain. We describe transaction malleability in detail in Section 4 followed by our proposed solution in Section 5. In Section 6, we provide a detailed analysis of our proposed protocol. In Section 7, we give our concluding remarks.

2. Related Work

Transaction malleability has been known about since around 2011 [13] but has been largely treated as a minor inconvenience [14]. Not only has transaction malleability been used to steal Bitcoins but before

that it was also used as a DDoS attack where attackers used transaction malleability to launch the DDoS attack in order to temporarily disrupt the balance-checking services of BitStamp, which handled the world's biggest volume of Bitcoin trading [15]. Wuille [16] proposed a Bitcoin Improvement Proposal (BIP) called bip-0062. In this BIP, he mentioned several sources of transaction malleability, such as Non-DER encoded ECDSA signatures, non-push operations in *scriptSig*, and inherent ECDSA signature malleability (just to name a few). The author of bip-0062 proposed new rules to eliminate transaction malleability. These rules enforce procedures that should be adopted in order to eliminate the above-mentioned sources of transaction malleability. Most of these rules apply to version 3 blocks. The problem with this solution is that it proposes a hardfork change that would make version 2 blocks entirely invalid when 95% of the past 1,000 blocks are version 3 or higher. Andrychowicz and Dziembowski [17,18] presented a solution for transaction malleability by eliminating the parts of the transaction that can cause malleability. They presented their approach for Bitcoin contracts. They proposed removing an important script from the transaction when the hash (transaction ID) is computed. The drawback of their approach is the removal of this important information that is signed by the owner. Decker and Wattenhofer [19] presented transaction malleability and the Mt. Gox issue. They analyzed various transactions in Blockchain to identify double spending attempts and tried to investigate if the attackers used malleability before the Mt. Gox incident. The authors only presented a case study and did not present a solution. The approaches in [20,21] studied the attacks based on double spending and transaction malleability. As we mentioned earlier, at the time of this writing, almost 361,290 blocks have been processed and added to Blockchain. A change in the protocol, such as presented in [16,18], might not be accepted broadly by the system participants due to its hardfork nature.

3. Bitcoin Basics

In this section we discuss the basics of Bitcoin to understand how transactions work in Bitcoin.

3.1 Transactions

Bitcoin transactions are signed scripts that are broadcast on the network [22]. A transaction mentions the transfer of funds from one Bitcoin address to another. A Bitcoin address is actually a hash of a public key. A client installs Bitcoin wallet software and generates an elliptic curve digital signature algorithm (ECDSA) public/private key pair. The public key is then hashed and serves as the address Bitcoins are transferred from/to. A transaction is written in Bitcoin scripting language. A transaction can be divided into two segments. One is the input that mentions the source of Bitcoins that are to be spent. It is specifically a reference to the output in some other transaction. There can be more than one input in a transaction mentioning unspent outputs in some other transaction(s). The total input value describes the sum of Bitcoins that are to be transferred. This input section also contains the hashes of those other transactions as a reference. One of the most important fields it contains is the *scriptSig*. The *scriptSig* contains some signed information along with the public key so that miners can verify the signature on the script. Along with the signature and the associated public key (the hash of which is mentioned in the output of previous transactions) the creator of this transaction proves their ownership

of the Bitcoin address containing the Bitcoins that are being spent. One noticeable thing is that all the Bitcoins mentioned in the input section must be spent (including the fees to the miners). Therefore, if the total number of Bitcoins contained in these referenced inputs is 100 Bitcoins and the redeemer wants to spend 75 Bitcoins then he/she needs to mention another output in his/her transaction, which contains his/her own address. This address serves to collect the change of 25 Bitcoins. The output segment, on the other hand, mentions the number of outputs carried by this transaction. For example, if the total number of Bitcoins is to be sent to two addresses then this field will be set to a value of 2. The output also contains the total number of Bitcoins needed to be sent, including the transaction fees, to the miners. The most important field in the output segment is the s*criptPubKey*, which is the other part of the script. It contains the destination address where the Bitcoins are to be sent, along with the instructions (*script*) that specify the conditions that must be fulfilled by the receiver in order to redeem this transaction. Upon receiving the transaction, the miners will execute this script to verify the transaction. Fig. 1 shows a typical transaction script denoted as the *Tx_script*. Note that the *scriptSig* field in the *Tx_script* is first temporarily filled with the *scriptPubKey* of the transaction whose output is to be redeemed. This *scriptPubKey* provides important information. For example, it mentions the conditions that were fulfilled in order to redeem the Bitcoins now contained in the previous transaction. This is particularly important in more advanced transactions, such as Bitcoin contracts. After the writing of this transaction is completed, the hash is taken and signed as shown below:

$$Sig\_data = (h(Tx\_script))_{SK} + hash\_type$$

where, $h(.)$ represents a one-way hash function, such as SHA-256, and *hash_type* is SIGHASH_ALL (a Bitcoin scripting word that means to calculate the signature on all of the script). This information is then signed by the owner's private key (SK).

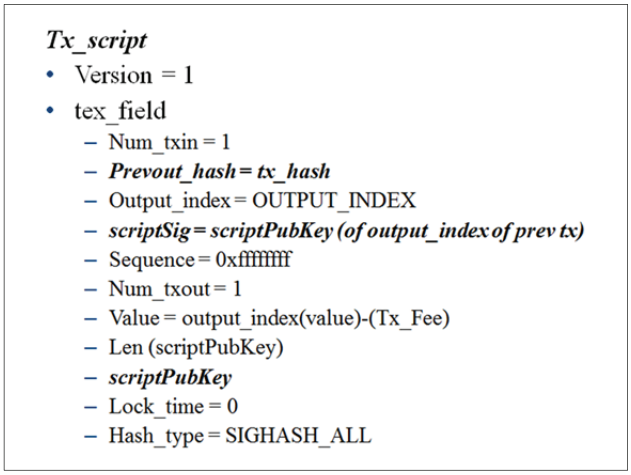

Fig. 1. A typical Bitcoin transaction script.

Note that some of the terms, such as *Tx_script*, in the above and subsequent equations are not necessarily a standard representation but are used in this paper in the attempt to provide a better understanding of the concept.

The Bitcoin address that contains the unspent Bitcoins is the hash of the corresponding public key of this private key. Finally the *scripSig* field is constructed as:

$$scriptSig = len(Sig\_data) + Sig\_data + len(pubkey) + pubkey$$

where, *len* denotes the length of the corresponding field in bytes and *Sig_data* is an ECDSA signature pair (*r,s*).

Note that these values in *scriptSig* are not signed, as it is unfeasible for a signature to sign itself. The final transaction is almost the same as *Tx_script* except that the *scriptSig* field is replaced by the newly created *scriptSig* value. The final transaction (*Final_Tx*) is shown in Fig. 2. Before relaying it to the network, a hash is computed over *Final_Tx* and serves as the transaction's unique identity (*tx_id*), as shown below:

$$tx\_id = h\ (Final\_Tx)$$

Note that this *tx_id* is used to search a particular transaction in the Blockchain. As we will see later in our paper, this *tx_id* plays a significant role in transaction malleability.

3.2 Blockchain

Bitcoin uses a database, known as Blockchain, for keeping record of all transactions. Blockchain contains every transaction that is ever executed in Bitcoin. With the help of this information, anyone can check about how many Bitcoins were associated with an address at any point in the past [23]. Blockchain is indeed a chain of the blocks containing transactions. Each block contains the hash value of the previous block. This chain is continued from the very first block, which is known as the genesis block. The hash values guarantee that this block is generated after the previous block. It is interesting to note that it is very difficult to modify a block after it has been verified. This is because after a while, there will be more verified blocks and if someone tries to change the block then he/she also needs to change the subsequent blocks as well.

A block is verified with the concept of the proof of work. A miner who is verifying a block containing transactions will need to verify all of the transactions. Besides this, the miner also requires to solve a mathematical challenge termed as *proof of work*. This mathematical challenge has been made to be difficult enough so that it requires some computing effort in order to be solved. The Bitcoin network provides miners with a challenge in order to find the proof so that when they take the hash of the puzzle it should result in a hash with certain number of leading zeros. The challenge is composed of the hash of the transactions (contained in the block), the previous hash of the block, and some nonce. In case of not finding the particular hash, miners increment the nonce and try again until they find the required number of leading zeros. The Bitcoin network continuously monitors the average time it takes for the miners to find this proof. On average, the time to solve the challenge should be around 10 minutes. As the mining hardware is becoming more and more sophisticated every day, this challenge needs to be made harder. For example, if the required number of zeros is 40 then on the average it requires $2^{40} = 1$ trillion hashes to be checked in order to find one such hash. If the Bitcoin network detects that the challenge being solved is taking less than 10 minutes on average then the difficulty of the challenge is increased by increasing the leading zero condition by one more bit, say 2^{41}, which equals 2 trillion

attempts on average to produce one required hash. This proof of work guarantees that if an attacker tries to insert a fake block (or a block containing a double spending transaction) then it needs to catch this proof of work with a computing power of 51% of the network's nodes.

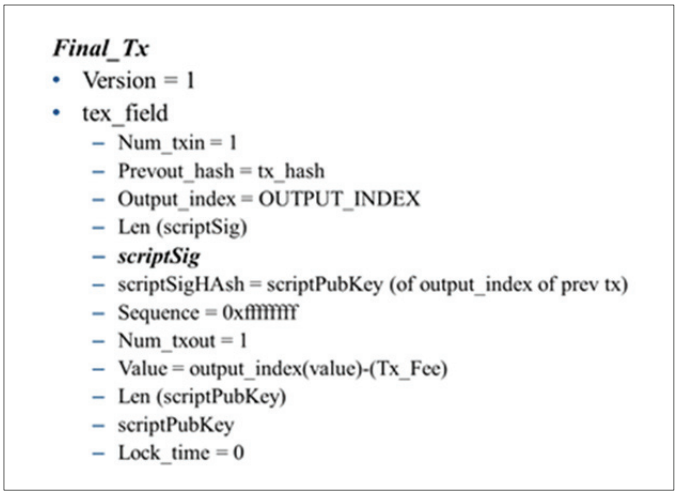

Fig. 2. Final transaction.

3.3 Mining and Miners

Miners are the nodes in the Bitcoin network who verify the blocks. Once a transaction is relayed across the network, it is combined in the pool of unconfirmed transactions. Miners collect these unconfirmed transactions in a block and start to verify it. Today, miners are equipped with specialized mining hardware, such as FPGA or ASIC-based mining devices. It is worth mentioning that the proof of work requires the calculation of hashes. These devices typically operate in speeds of Giga Hash (Ghash). Once the solution to the challenge is found, the miner(s) relay(s) this solution across the network. The other miners verify this solution. Once verified, this block is added to Blockchain and the miners start working on the next block. The miner(s) who solves the block is/are rewarded with newly generated Bitcoins, which are currently set to 25 Bitcoins. For doing so, the miners create a special transaction known as a "coinbase" in which they send 25 Bitcoins to themselves.

3.4 Scripts

Bitcoin scripts play a fundamental role in Bitcoin transaction malleability. A Bitcoin *script* is essentially a series of instructions in a transaction that are written to describe how the Bitcoins containing in this transaction will be redeemed. A *script* governs the conditions that must be fulfilled by the spender. A standard Bitcoin transaction requires the spender to provide two things: The public key that hashes to the address that contains the Bitcoins to be spent, and the signature, which ensures that the spender has the corresponding private key of the public key. The commands or functions in a script are called *script words* and are stored on a stack upon execution. A transaction will be considered valid if the top stack item returns true, otherwise it will not be a valid transaction. Certain words are used to push byte data onto the stack. Some are used for flow control; some for bitwise logic operation;

arithmetic operations; some for crypto operations, such as taking the hash; and some for checking the signature. Below is the standard transaction script that is used to transfer Bitcoins to a public key hash address.

scriptPubKey: OP_DUP + OP_HASH160 + <pubKeyHash> + OP_EQUALVERIFY
+OP_CHECKSIG
scriptSig: <sig> < pubkey>

The above script typically means to duplicate the top item of the stack (public key), take its hash (*pubkey hash*), compare it with the *pubKeyHash*, which is also presented in the script, and verify it. On returning true, the script is considered to be validated. *scriptSig* contains the signature and the public key. Another important thing related to words is their representation in hex codes in a raw transaction. For example, 76 is the hex code for *OP_DUP*, *A*9 is the hex code for *HASH160*, and so on. The "raw transaction API" was developed to give developers or very sophisticated end users access to creating and broadcasting transactions. Thus, it allows a company to create its own software according to its need to exchange Bitcoins. More precisely, it gives them the freedom to exchange Bitcoins in the way their business operates. Finally, a transaction is identified by its hash, which is a very important characteristic of Bitcoin protocol. Once a raw transaction is written and dispatched over the network, it will be identified in Blockchain by its hash. Therefore, once a complete transaction is written, mentioning all the inputs, outputs, *scriptSig*, *scriptPubKey*, etc., the hash of this final transaction serves as the transaction ID.

4. Transaction Malleability

In this section we explain transaction malleability and how it works. This bug was already present in the system and first identified around 2011 [9]. Until recently it was not considered to be a very serious issue. In this section we describe two of the incidents that involved transaction malleability.

Mt. Gox was a Tokyo-based Bitcoin exchange. It was established in 2010 and by February 2014, it accounted for almost 70% of all Bitcoins ever traded [19]. In February 2014, this company suspended its trading, closed its website, and filed for bankruptcy protection [24]. As it was working as a Bitcoin exchange, people used to deposit other currencies like US dollars in exchange for Bitcoins. The attackers launched a transaction malleability attack by requesting a transfer of Bitcoins from Mt. Gox and then intercepting and replacing that transaction with a modified one with a different hash. Because the modified transaction still contained all of the valid addresses and the public key address, the network verified it and the Bitcoins were transferred to the attacker's account. On the other hand, because its ID had been changed, the attacker complained to the company that he/she had not received the intended Bitcoins and the company re-sent the Bitcoins. Mt. Gox later announced that they were affected by this bug in Bitcoin software and lost around 850,000 Bitcoins. However, later they said that they recovered some of them. Their total loss was more than 450 million US dollars. Bitcoin developers said this was a known bug and Mt. Gox had to have exchange transaction software that was in accordance with this bug. Another Bitcoin trading place, Silk Road 2, who lost over 4,000 Bitcoins that were valued at approximately 2.7 million US dollars, reported a similar case.

4.1 Types of Transaction Malleability

We know that when the hash of some data is produced, then even a slight change results in a completely different hash. As mentioned earlier, the hash of the final transaction serves as the transaction ID. If someone is able to change some of the details in the transaction, the resultant hash will be completely different. Thus, we can define transaction malleability as an attack in which an attacker can change the unique transaction ID by manipulating the transaction, which results in a different hash while all the other details of the transaction remain the same. This particularly means that the miners will verify the transaction. This leads to the question of: "What can an attacker change in a signed and hashed transaction?" A transaction includes components like the receiver's address, amount, inputs, etc. To protect this data, the hash of a certain part is calculated and signed. This signed information is kept in *scriptSig* as a part of the final transaction. Note that this signature is calculated over only a part of the transaction and the complete final transaction is not signed. That is, the *scriptSig* does not sign itself, as it is unfeasible for a signature to sign itself. This introduces two ways of inducing malleability [25], as described below.

4.1.1 Signature malleability

The first type of malleability is related to the signature. Bitcoin uses an ECDSA signature. For every ECDSA signature $(r, s(modN))$, the signature $(r, -s(modN))$ is also a valid signature of the same message. Therefore, if an attacker intercepts the transaction and changes the sign of s in the signature pair (this modification is possible due to the fact that the signature is not signed) then, the resultant hash (the transaction ID) will be completely different. Second, each signature has only one DER encoded ASN.1 octet representation. However openssl, which is used in the Bitcoin network, does not enforce it unless the signature is too ill formed [25]. This means that if an attacker pads the signature with $0x00...$ *or* $0xFF....$ then the signature will be accepted and the resulting modified transaction will contain a completely different hash.

4.1.2 Script malleability

Script malleability is related to script words. We know that *scriptSig* is not signed in a transaction. This allows an attacker to add certain script words, such as *OP_DROP*, to the scripts. *OP_DROP* simply leaves the stack as before, and, therefore, its use will not affect the execution of a script. This addition in a transaction script will result in a completely different hash and a different transaction ID. There is another way of adding or changing op codes. As script words are represented with op codes in hex format, one can change the op codes in such a way that two different op codes (script words) work in a similar manner. For example, *PUSHDATA* 48 means push two bytes of data and *OP_PUSHDATA2* means the next two bytes. Because these two different op codes are doing exactly the same thing, it will not change the functionality of the transaction. However, two transactions with different op codes will have completely different hashes (or transaction IDs).

This is what happened to Mt. Gox [19]. Although the expected transaction IDs of their transactions were not appearing in Blockchain, they were being executed and their clients (the attackers) were getting Bitcoins. The attackers were claiming that they were actually not receiving Bitcoins and Mt. Gox software was resending payments. By the time they knew what had happened, it was already too late.

5. Proposed Solution

This section provides our proposed solution, the structure of our proposed transaction ID, and the proof of the concept.

5.1 Proposed Transaction ID

A hash has a fixed length. Therefore, all of the transactions, whether they are unmodified or malleable transactions, have the same size hash. The transaction ID serves as a reference in a subsequent transaction that tries to redeem the unspent Bitcoins in it. Therefore, the modification or removal of this information from the transaction script will be a hardfork and will need a major change in protocol. Thus, it is desirable to keep the current structure of the transaction script. The idea of our solution is to use the hash of the transaction on which the signature is calculated (i.e., $h(Tx\_script)$). That means, we aim to use the hash that is calculated without the input script (i.e. *scriptSig*). Note that the values of this hash cannot be modified and any change to *scriptSig* in *Final_Tx* will not modify this hash. We propose that this hash should be appended with the hash of the final transaction to be used as the transaction ID. We refer to this as $hash_{new}$. Our proposed format for the transaction ID will look like $hash_{prev} \,||\, hash_{new}$, which is shown in Fig. 3, where $hash_{prev} = h(Final\_Tx)$, which serves as the transaction ID in the current protocol. By doing so, we are only slightly modifying the current protocol for only the identification requirement of the transaction. In other words, because our new transaction ID has two fixed length segments, miners will always take the first part of the transaction (i.e., $hash_{prev}$) to verify it as they did before. During the transaction verification procedure, the miners will validate the transaction in a way that similar to the existing protocol. This includes the validation of $hash_{new}$ during the signature verification.

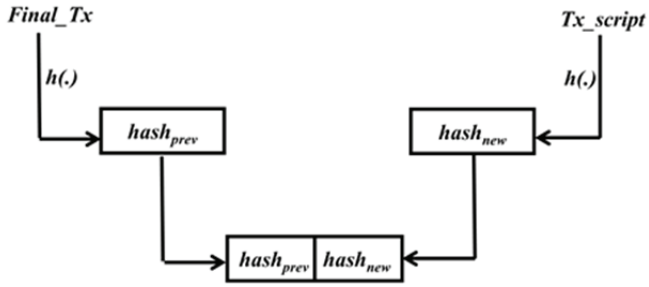

Fig. 3. The new structure of transaction ID.

1. Each of the transactions that are referencing the unspent outputs of this transaction will keep referencing the transaction with $hash_{prev}$. Even if the $hash_{prev}$ is modified, it will cause no harm and will continue to serve as the transaction reference in subsequent transactions that are referencing it.
2. A sender, who tries to find his/her transaction in Blockchain, will look for the $hash_{new}$ and will eventually find it. Thus, it solves the problem of the sender who is not finding his/her transaction in Blockchain. Furthermore, the sender will also be able to detect any malleability attempt as he/she will find the corresponding $hash_{prev}$.

We will further demonstrate our approach in the security analysis section.

5.2 Proof of the Concept

In this section we provide the proof of the concept of our scheme.

Lemma 1. If an attacker successfully changes $hash_{new}$ in the proposed transaction ID.

Proof. As mentioned in our proposed transaction ID:

$$hash_{new} = h(Tx\_script)$$

Let's suppose an attacker changes the *Tx_script* in order to modify $h(Tx\_script)$,

$$hash_{new'} = h(Tx\_script')$$

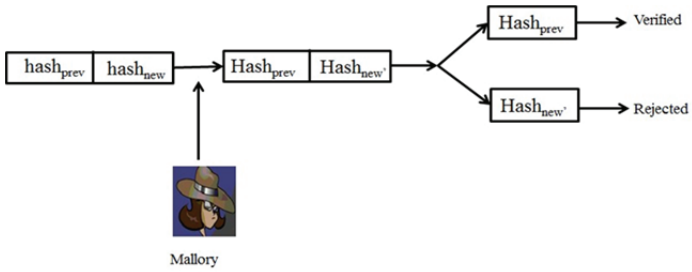

Fig. 4. Showing attempt of modification of $hash_{new}$.

But we know that *Final_Tx* contains the signature pair (r,s) that is calculated on $h(Tx\_script)$ so,

$$hash_{new'} \neq hash_{new}$$

Therefore, the transaction will be invalidated, as shown in Fig. 4.

Lemma 2. If an attacker successfully changes $hash_{prev}$ in the proposed transaction ID.

Proof. As mentioned in our proposed transaction ID:

$$hash_{prev} = h(Final\_Tx)$$

Let's suppose an attacker changes the *Final_Tx* in order to modify $h(Final\_Tx)$ with the help of transaction malleability, as explained in Sections 4.1 and 4.2.

$$hash_{prev'} = h(Final\_Tx')$$

But we know that a change in the *Final_Tx* does not affect the functionality of a transaction. Moreover, once verified, $hash_{prev'} \,||\, hash_{new}$ is searchable by the sender.

Therefore, a changed $hash_{prev'}$ is detectable, as shown in Fig. 5.

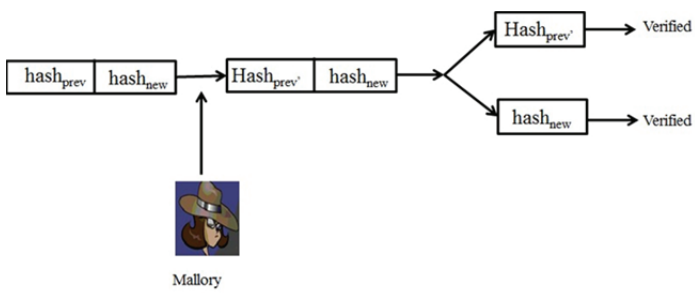

Fig. 5. Showing attempt of modification of $hash_{prev}$.

6. Analysis

In this section we provide a comparison with existing work, as well as the communication and computational overhead of our proposed scheme.

6.1 Comparison with Current Approaches

Recently, Wuille [16] proposed a BIP known as bip-0062 that states several sources of transaction malleability to make *scriptSig* malleable. They also proposed new rules to eliminate transaction malleability. However, most of these rules apply to version 3 blocks, which means that the hardfork nature of their solution is clearly a drawback.

Another solution proposed by Andrychowicz and Dziembowski [17,18] is based on a solution that involves removing input scripts, which contain information about how a previous transaction was redeemed, from the transaction when the hash (transaction ID) is computed. The drawback of their approach is the removal of the important information that is signed by the owner.

Compared to these schemes, it is evident that our solution preserves the important information contained by *scriptSig* in *Tx_script* and does not require a hardfork change in the current protocol. Table 1 shows a comparison with other current approaches.

Table 1. Comparison with current approaches

Parameter	Wuille [16]	Andrychowicz and Dziembowski [17,18]	Proposed solution
Requires Hardfork	Yes	Yes	No
Requires *scriptSig* removal	No	Yes	No

6.2 Computational, Communicational, and Storage Overhead

Lemma 3. In our proposed solution there are not any extra computational costs for miners in verifying a transaction based on our proposed solution.

Proof. Since the miners already verify *Tx_script* by removing the *scriptSig* from *Final_Tx*, they automatically verify $hash_{new}$. This is the most promising feature of our proposed approach that does not require any extra effort by miners in verifying the new transaction ID.

Lemma 4. The proposed solution only requires 256 additional bits to be communicated over the network and stored in Blockchain.

Proof. $hash_{new}$ is the only extra addition in our protocol. A SHA-256 hash is 256 bits. This will be appended with the hash of the final transaction before transmission. Similarly, in Blockchain, each transaction will now be identified by not only the $hash_{prev}$, which is currently being used, but also with the 256-bit $hash_{new}$. We argue that this is a reasonable tradeoff as it eliminates the problem of transaction malleability.

7. Conclusions and Future Work

Transaction malleability has caused serious damage to the Bitcoin community in terms of the loss of money and reputation. Not only were millions of dollars worth of Bitcoins stolen, but also the value of Bitcoin has been declining since this issue. We have presented a thorough study related to Bitcoin transactions and various attacks based on transaction malleability. We have proposed a robust mechanism to cater to this issue. Our proposed solution is comprised of a transaction ID that is a combination of the hash of the transaction script without the signature and the hash of the final transaction. This combined hash serves for the purposes of both transaction verification and identification. We analyzed our approach with respect to possible attacks presented in the proof of the concept, as well as computational, communicational, and storage overhead, and we found that our solution is feasible and detects any attempt of transaction malleability. Furthermore, it does not incur computational overhead, while only a small amount of communicational and storage overhead is required. We conclude that our approach is practical and in compliance with the existing Bitcoin protocols. In the future, we aim to implement our protocol to evaluate it further with respect to various overheads involved.

Acknowledgement

This work was supported by the Korea National Research Foundation (NRF) grant that is funded by the Korean Ministry of Education, Science and Technology MEST (No. NRF-2015R1D1A1A09058200).

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2014-1-00636) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

[1] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008; https://bitcoin.org/bitcoin.pdf.

[2] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Financial Cryptography and Data Security*. Heidelberg: Springer, 2013, pp. 34-51.

[3] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proceedings of the 13th ACM Internet Measurement Conference*, Barcelona, 2013, pp. 127-140.

[4] Bitcoin network data; http://www.coindesk.com/data/bitcoin/.

[5] How bitcoins mining works, http://www.coindesk.com/information/how-bitcoin-mining-works.

[6] Y. Takemoto and S. Knight, "Mt. Gox files for bankruptcy, hit with lawsuit," 2014; http://www.reuters.com/article/us-bitcoin-mtgox-bankruptcy-idUSBREA1R0FX20140228.

[7] A. Greenberg, "Silk Rod 2.0 'Hack' blamed on bitcoin bug, all funds stolen," 2014; http://www.forbes.com/sites/andygreenberg/2014/02/13/silk-road-2-0-hacked-using-bitcoin-bug-all-its-funds-stolen/#788200274a51.

[8] D. Bradbury, "Silk Road 2 loses over 2.33€ million in bitcoins in alleged hack," 2014; http://www.coindesk.com/silk-road-2-loses-bitcoins-hack.

[9] Bitcoin Foundation, "Contrary to Mt. Gox's statement, bitcoin is not at fault," 2014; http://bitcoinfoundation.org/contrary-to-mt-goxs-statement-bitcoin-is-not-at-fault/.
[10] Hardfork definition, https://en.bitcoin.it/wiki/Hardfork.
[11] Softfork definition, https://en.bitcoin.it/wiki/Softfork.
[12] U. Rajput, F. Abbas, R. Hussain, H. Eun, and H. Oh, "A simple yet efficient approach to combat transaction malleability in bitcoin," in *Information Security Applications*. Cham: Springer International Publishing, 2014, pp. 27-37.
[13] New attack vector, https://bitcointalk.org/index.php?topic=8392.msg122410#msg122410.
[14] K. Shirriff, "Bitcoin transaction malleability: looking at the bytes," 2014; http://www.righto.com/2014/02/bitcoin-transaction-malleability.html.
[15] M. J. Schwartz, "Bitcoin exchanges buckle under DDoS attack," 2014; http://www.darkreading.com/attacks-and-breaches/bitcoin-exchanges-buckle-under-ddos-attacks/d/d-id/1113809.
[16] P. Wuille, "Dealing with malleability," 2014; https://github.com/bitcoin/bips/blob/master/bip-0062.mediawiki.
[17] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Fair two-party computations via bitcoin deposits," in *Financial Cryptography and Data Security*. Heidelberg: Springer, 2014, pp. 105-121.
[18] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "How to deal with malleability of bitcoin transactions," 2013; http://arxiv.org/pdf/1312.3230v1.pdf.
[19] C. Decker and R. Wattenhofer, "Bitcoin transaction malleability and MtGox," in *Computer Security – ESORICS 2014*. Cham: Springer International Publishing, 2014, pp. 313-32.
[20] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, Raleigh, NC, 2012, pp. 906-917.
[21] M. Rosenfeld, "Analysis of hashrate-based double-spending," 2012; https://bitcoil.co.il/Doublespend.pdf.
[22] Transaction definition, https://en.bitcoin.it/wiki/Transactions.
[23] Bitcoin Block Explorer, https://blockchain.info.
[24] MtGox, "Notice of the results of approval or disapproval," 2014; http://www.mtgox.com/pressrelease20140210.html.
[25] Transaction malleability definition, https://en.bitcoin.it/wiki/Transaction_Malleability.

Ubaidullah Rajput https://orcid.org/0000-0002-3621-6458

He received his bachelor's degree in Computer System Engineering from Quaid-e-Awam University of Engineering, Science and Technology (QUEST), Pakistan in 2005. He received his master's degree in Computer System Engineering from NUST Islamabad, Pakistan in 2011. He successfully completed his Ph.D. in Computer Engineering from Hanyang University, Korea in 2017. His research interests are security and privacy issues in crypto-currency, security and privacy issues in VANETS, Internet of Things (IoT), mobile social networks and cloud computing. He has more than 12 years of teaching and research experience and currently working as Assistant Professor in QUEST, Pakistan. He has served as a reviewer in many conferences and journals. He is author of many international and national papers.

Fizza Abbas https://orcid.org/0000-0002-1346-335X

She received the bachelor's degree in computer system engineering from the Quaid-e-Awam University of Engineering, Science and Technology (QUEST), Pakistan, in 2007, and the master's degree in communication system and networks from Mehran University, Pakistan, in 2011. She successfully completed her Ph.D. in Computer Engineering from Hanyang University, Korea in 2017. Her research interests are security and privacy in social network services, mobile social networks, cloud computing, mobile cloud computing, and vehicle ad hoc networks. She has more than ten years of teaching experience and currently working as Assistant Professor in QUEST, Pakistan. She has served as a reviewer in many conferences and journals. She is author of many international and national papers.

Heekuck Oh https://orcid.org/0000-0002-2989-8737

He received his B.S. degree in Electronics Engineering from Hanyang University in 1983. He received his M.S. and Ph.D. degrees in Computer Science from Iowa State University in 1989 and 1992, respectively. In 1994, he joined the faculty of the Department of Computer Science and Engineering, Hanyang University, ERICA campus, where he is currently a professor. His current research interests include network and system security. Prof. Oh is President Emeritus of Korea Institute of Information Security & Cryptology, and is a member of Advisory Committee for Digital Investigation in Supreme Prosecutors' Office of the Republic of Korea. He is also a member of Advisory Committee on Government Policy under Ministry of Government Administration and Home Affairs. He is the corresponding author of this paper.