My two Bitcoins? Implementation of Double-Spending on Fast Bitcoin Payments

Juan David Peña Melo[2], Saúl Eduardo Pomares Hernández[1,2], Lil María Xibai Rodríguez Henríquez[3], and Julio César Pérez Sansalvador[3]

[1] LAAS-CNRS, Université de Toulouse, Toulouse F31400, France
[2] INAOE, Santa María Tonantzintla, Puebla, 72840, México
[3] INAOE-Cátedra CONACyT, Santa María Tonantzintla, Puebla, 72840, México

Abstract. Bitcoin is a payment system that eliminates trusted intermediaries in the exchange of digital currencies. To process transactions, Bitcoin uses a set of nodes with different specialized roles that function as a trusted third party. The Bitcoin confirmation transaction is a slow process that can take up to 72 hours. However, in fast payment scenarios, products are delivered immediately. These scenarios in Bitcoin are vulnerable to double-spending attacks. Different strategies have been proposed to mitigate the double-spending attack on Bitcoin, such as allowing transactions to propagate freely in the network, inserting a new node role to detect attacks, and penalizing malicious users for revealing their identities. To the best of our knowledge, there are no works to avoid double-spending attacks on fast Bitcoin payments. This article is a guide that shows how easy it is to perform double-spending attacks on fast Bitcoin payments and highlights the vulnerability of Bitcoin when the transaction is unconfirmed. The experiments run on the Bitcoin Testnet, an environment where coins are worthless, and developers can experiment on a distributed network infrastructure. The experiments show an analysis of the speed of Bitcoin to process transactions, the low investment that a malicious user needs to make to carry out the attack, and the high probability of success of a double-spending attack in fast Bitcoin payment scenarios.

Keywords: Bitcoin · Cryptocurrencies · Double-Spending · Peer-to-Peer · Fast-Payments

1 Introduction

Bitcoin is a payment system created by Satoshi Nakamoto that uses distributed systems and cryptography to eliminate trusted intermediaries in the value exchange [1]. Bitcoin does not have a central server that serves as a control point to process transactions and uses a set of nodes with different specialized roles that work as a trusted third party [2]. The confirmation of a transaction in Bitcoin is a slow process that can take up to 72 hours and is a process irreversible once the transaction is processed and confirmed in the Blockchain [3]. Bitcoin developers designed the payment system for internet sales where it is possible to wait for confirmation before delivering the product.

However, in fast payment scenarios, products are delivered immediately (in seconds order), for example, in ATMs [4] or takeaway restaurants [5]. These scenarios are vulnerable to Bitcoin double-spending attacks because the payment must be confirmed when the product or service is delivered, and the Bitcoin confirmation process is not fast enough, which increases the probability of successful double-spending attacks. In a successful double-spending attack, a malicious Bitcoin user pays twice with the same currencies, i.e., pays a seller and reverses the transaction so that the currencies go back to an address of his own [6].

Recently, have been proposals in the literature to mitigate double-spending attacks on fast Bitcoin payments. The first strategy is to propagate all transactions in the network without restrictions so that the network nodes can identify the double-spending attack [7] [8]. Another advance is to introduce observers to alert attack nodes [9]. A third approach avoids network isolation to ensure a higher probability of seeing inconsistencies related to double-spending attacks on the system [10]. Finally, another strategy is to reveal the identity of malicious users attempting double-spending attacks [11]. Currently, Bitcoin does not guarantee a complete solution for double-spending attacks on fast Bitcoin payments.

This article is a guide to a double-spending attack on fast Bitcoin payments. This guide is based on Karame's requirements for a successful double-spending attack [7] and shows in detail how a malicious user can take advantage of the distributed nature of the system to purchase products and services without spending their coins. The experiments run on the Bitcoin Testnet [12], an environment where coins are worthless, and developers can experiment on a distributed network infrastructure. The experiments show an analysis of the speed of Bitcoin to process transactions, the low investment that a malicious user needs to make to carry out the attack, and the high probability of success of a double-spending attack in fast Bitcoin payment scenarios.

2 Background

2.1 Bitcoin Overview

Bitcoin is a peer-to-peer payment system based on cryptography and distributed systems. The network's peers have various roles, such as mining nodes, full blockchain nodes, full nodes and lightweight nodes [13, p. 172]. The purpose of peers is to propagate, verify, and confirm transactions that transfer value between network users without needing a trusted entity.

Transactions are data structures cryptographically signed by the owner that can exchange value on Bitcoin. The data structure is composed of an identifier, a pointer to the previous transition called input, and outputs that define the new owners of the coins. To launch a transaction in Bitcoin it is necessary to connect through a node. Every time a node receives a transaction, it verifies that:

1. The transaction has enough Bitcoins to consume, i.e., the output must not exceed the input.
2. The input is spent once.
3. The digital signature is authentic.

Once the transaction is verified, the nodes store it in a memory space called a Mempool [14], where it waits for confirmation. The confirmation process refers to inserting a transaction into the Blockchain through "mining" a new block. The miners are a set of nodes that reach a consensus through a non-deterministic process to insert the new blocks in the Blockchain. The mining process is beyond the scope of this article, as a successful double-spending attack on fast Bitcoin payments does not need computational power.

2.2 Karame's Model

The Karame model consists of a malicious user and a seller connected to the Bitcoin network. The malicious user wants to buy a product from the seller without spending his coins. To achieve this, the malicious user carries out a double-spending attack. The double-spending attack concerns spending the same currency twice [7]. In Karame's model, the malicious user controls multiple nodes that help execute the attack since he cannot sign two transactions that spend the same currency on a single node. The malicious user does not have enough computational power to create a block, and a transaction belonging to a block is considered irreversible [8].

Necessary Conditions for Successful Double-Spending A successful double-spending attack is performed as follow: Alice creates two transactions that spending the same currency $Tr(A)$ and $Tr(B)$, the transaction $Tr(B)$ pays Bob for the product Alice wants to buy, while transaction $Tr(A)$ returns the coin to Alice. Notice that the double-spending attack is a competition in the propagation of the two transactions to belong to the Blockchain. To achieve a successful double-spending attack, Alice must ensure that $Tr(A)$ belongs to the Blockchain before transaction $Tr(B)$ by meeting the following requirements:

- Requirement 1 - $Tr(B)$ is added to Bob's Mempool. If $Tr(B)$ is not added to Bob's Mempool, then there is no product delivery because there is no evidence that Alice wants to pay Bob.

- Requirement 2 - $Tr(A)$ is confirmed on the Blockchain. If transaction $Tr(B)$ is confirmed before transaction $Tr(A)$, Alice will not be able to get her coins back and, Bob will receive payment for the product.

- Requirement 3 - Bob's product delivery time is less than Alice's misbehavior detection. Bob needs to be in a fast-payment scenario to ensure the product is delivered before the Blockchain confirmation.

3 Analysis of Bitcoin Transactional Processing

We analyze the Bitcoin Mempool with 30,000 transactions between blocks 753957 and 753965. We show this data in Figure 1, where gray dots are confirmed transactions and red dots are unconfirmed transactions. On the x-axis, we show the observation and data capture time, while on the y-axis, we see the amount of fee a transaction pays per byte. Finally, the colored bars represent the instant in time where a block is added to the blockchain.

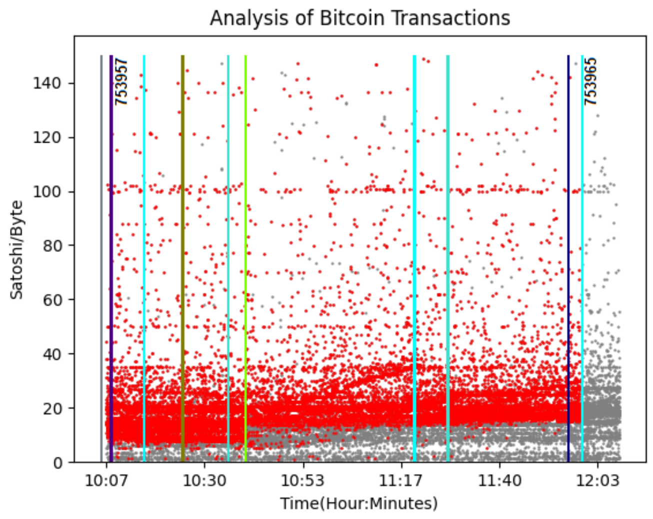


Fig. 1. Confirmation of transactions in the blockchain and stagnation of transactions in the Mempool. This graph can be generated with the repository by publishing in [15].

Analyzing Figure 1 we find that there are two scenarios where a malicious user can perform successful double-spending attacks:

First Scenario: in Bitcoin, each transaction pays a fee for the number of bytes added to the Blockchain. Fees are rewarding for miners to receive for the computational power invested in the network. Most nodes adjust the Satoshi/Byte fee (Satoshi is the smallest unit of a Bitcoin that can be sent, that is, hundredth of a millionth Bitcoin) based on the number of transactions in the Mempool [16]. Figure 1 shows from 10:30 to 10:53 GMT-5 the increase in fees for the number of transactions in the Mempool. Note that Bitcoin's transaction processing capacity is a bottleneck for the system, causing many transactions to remain on hold without confirmation for up to 72 hours or eventually be discarded [3]. During the propagation and confirmation time of a transaction, a malicious user has the advantage of performing a double-spending attack on Bitcoin, propagating two transactions with low-fee that spend the same currency and obtaining a product or service from a merchant who does not wait for the confirmation.

Second Scenario: Bitcoin creates a block every 10 minutes on average. However, Figure 1 shows that there are long time intervals in the creation of new blocks, i.e., the standard deviation is high, which means that sometimes there are time intervals of up to 40 minutes between one block and another [17]. This standard deviation also increases the risk of a successful double-spending attack in fast payment scenarios. Because if a malicious user spreads two transactions that spend the same currency with a high fee, he would still have the long confirmation times, which are sufficient in scenarios where the exchange of products and services is immediate.

4 Related Work

The double-spending attack on Bitcoin fast payments is a vulnerability detected by Karame et. al in [7] [8]. The author identifies the necessary requirements for a successful double-spending attack based on his Bitcoin fast-payment model. They propose that all transactions, including inconsistent ones, propagate without restrictions in the network so that every node can detect double-spending attacks. However, the mechanism proposed by Karame could generate inconsistencies in the network for a prolonged period, affecting the consensus between the nodes and causing a denial of service attacks.

Based on the fast payments of Bitcoin and the vulnerability proposed in the Karame model, T. Bamert et. al in [10] propose a strategy to identify double-spending attacks by connecting to random nodes and listening for transactions in the network that have inconsistencies. This solution can identify double spending attacks with a high percentage as long as the set of nodes is permissioned and limited. However, the Bitcoin network is permissionless, and its pool of nodes tends to grow over time.

C. Pérez-Solà et. al in [11] proposes to mitigate the double spending attack by penalizing malicious users. For this, it uses a vulnerability of the digital signature scheme based on the elliptic curve to reveal the private key of the attackers. However, revealing the users' private key in an asymmetric encryption scheme is undermining the security foundation of Bitcoin, and this would cause a risk to possibly not malicious users.

The works mentioned before are the most relevant in the literature on double-spending attacks on fast Bitcoin payments. However, the proposed strategies do not guarantee 100% attack detection, opening a topic for future research.

5 Double-spending attacks on Bitcoin Testnet

In Bitcoin, transactions have no temporal constraints, and the confirmation process starts when a transaction is added to a block. Bitcoin payments are slow and refer to a scenario where the merchant waits for up to 6 confirmations from the Blockchain to deliver the product. However, the double spending attacks shown in this section are based on Karame's model, they set up a scenario of fast payments without waiting for any confirmation to deliver the product, and this scenario runs our attacks.

The section is organized as follows: first, the differences between the Bitcoin Mainnet and the Testnet. Second, the hardware and software used in the attacks. Third, a description of the transactions that spend the same currency, then the attacks are described, and finally, a discussion about the results found.

5.1 Difference between Bitcoin Testnet and Bitcoin Mainnet

Bitcoin has two disjoint Blockchains: Bitcoin Mainnet and Bitcoin Testnet. Developers used Bitcoin Testnet as environment to test without spending money or causing inconsistencies in the Bitcoin Mainnet. Coins on Testnet are separate from real Bitcoins and never have value. The differences are shown below:

Bitcoin Testnet	Bitcoin Mainnet
No monetary value	Real value
The difficulty restarts	The difficulty is variable
Port 18333	Port 8333
Transaction Frequency Low	Transaction Frequency High
No economic benefit for mining	Economic incentive for mining
The data is periodically deleted	Traceability from the Genesis Block
Connection Port RPC 18332	Connection Port RPC 8332

Table 1. Bitcoin Testnet vs Bitcoin Mainnet

The transactional verification process, from creating a transaction to adding it to the blockchain, is similar on both Blockchains [13]. Therefore, a double-spending attack on fast Bitcoin payments runs the same on Testnet and Mainnet. However, since the frequency of transactions is higher on the Mainnet, the fee for each transaction is more expensive. The scenario posed in the Karame model [8] refers to a fruit seller who receives fast Bitcoin payments. This scenario is not profitable for a malicious user since executing a double-spending attack on the Bitcoin Mainnet would be more expensive than the product. However, not all fast payment scenarios handle low amounts. For example, an ATM [4] exchanging Bitcoin for cash can be a high-money-loss scenario if a malicious user achieve a successful double-spending attack, as shown in section 5.4 Figure 8.

5.2 Hardware and Software

We perform double-spending attacks on Bitcoin Testnet [12] following Karame's model with the two scenarios seen before. The hardware used for these experiments is an Orange pi PC [18] minicomputer show in Figure 2, under the Armbian Buster [19] operating system. This minicomputer is chosen for its low cost,

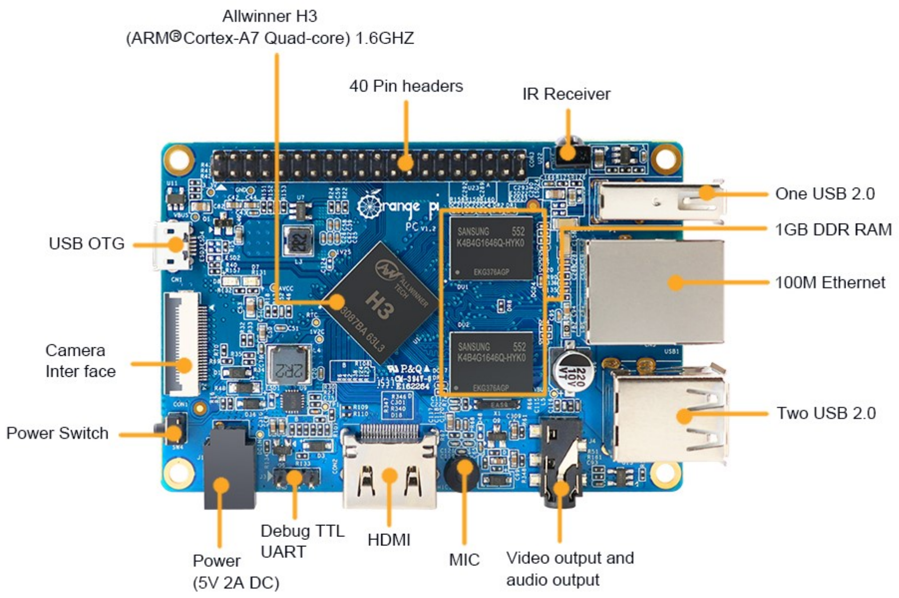


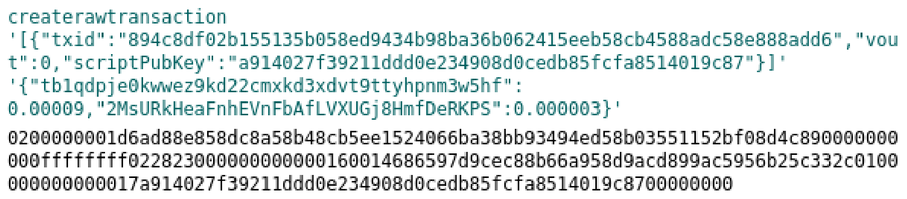
Fig. 2. Orange Pi PC minicomputer with hardware specifications [18].

and power consumption. It also meets technical specifications that require a Bitcoin node to store the Blockchain.

For the implementation of the attacks, the Bitcoin Core software is installed on 3 Orange pi PC in the role of lightweight nodes. The nodes simulate the behavior of a malicious entity named Alice, an entity merchant named Bob, and an Alice's Helper Node. The process of synchronizing the nodes with the network takes up to 3 hours. Subsequent, each node must be able to send and receive Bitcoin transactions via a generated public address and private key. The addresses are generated in the Bitcoin Core console with the `getnewaddress` command. The public key is similar to a bank account number and is used to receive transactions. The private key signs transactions to be propagated on the Bitcoin network.

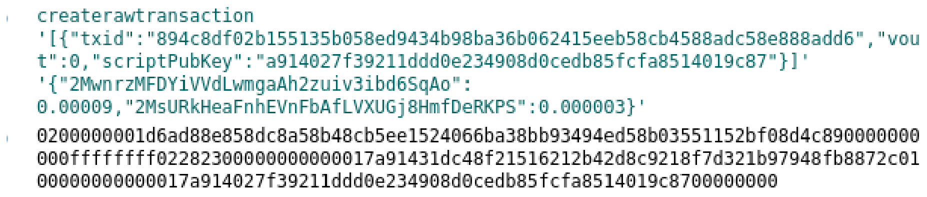
5.3 Create and propagate Bitcoin transactions that spend the same currency

Bitcoin transactions are created by pointing to the identifier `txid` of the previous transaction and the output to be consumed. We assume that Alice makes two transactions pointing to the same identifier `txid` and the same output. $Tr(A)$ is a malicious transaction that returns the coins to Alice, and $Tr(B)$ is a transaction that tries to pay Bob for the product. Figure 3 and Figure 4 show the creation of two transactions $Tr(A)$ and $Tr(B)$.



```
createrawtransaction
'[{"txid":"894c8df02b155135b058ed9434b98ba36b062415eeb58cb4588adc58e888add6","vou
t":0,"scriptPubKey":"a914027f39211ddd0e234908d0cedb85fcfa8514019c87"}]'
'{"tb1qdpje0kwwez9kd22cmxkd3xdvt9ttyhpnm3w5hf":
0.00009,"2MsURkHeaFnhEVnFbAfLVXUGj8HmfDeRKPS":0.000003}'
0200000001d6ad88e858dc8a58b48cb5ee1524066ba38bb93494ed58b03551152bf08d4c890000000
000ffffffff0228230000000000000160014686597d9cec88b66a958d9acd899ac5956b25c332c0100
000000000017a914027f39211ddd0e234908d0cedb85fcfa8514019c8700000000
```

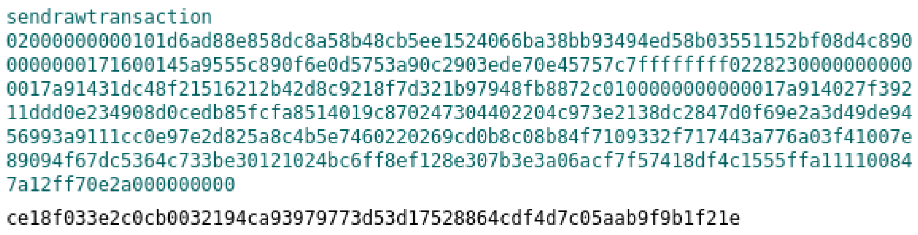
Fig. 3. Creation of the malicious transaction Alice $Tr(A)$.



```
createrawtransaction
'[{"txid":"894c8df02b155135b058ed9434b98ba36b062415eeb58cb4588adc58e888add6","vou
t":0,"scriptPubKey":"a914027f39211ddd0e234908d0cedb85fcfa8514019c87"}]'
'{"2MwnrzMFDYiVVdLwmgaAh2zuiv3ibd6SqAo":
0.00009,"2MsURkHeaFnhEVnFbAfLVXUGj8HmfDeRKPS":0.000003}'
0200000001d6ad88e858dc8a58b48cb5ee1524066ba38bb93494ed58b03551152bf08d4c890000000
000ffffffff02282300000000000017a91431dc48f21516212b42d8c9218f7d321b97948fb8872c01
00000000000017a914027f39211ddd0e234908d0cedb85fcfa8514019c8700000000
```

Fig. 4. Creation of the transaction that trying to pay Bob $Tr(B)$.

Note that in Figures 3 and 4, transactions point to the same identifier txid and have a different recipient, i.e., the two transactions spend the same currency. After the transactions are created and signed, be propagated on the network. The propagation of the transactions to the network is done through the sendrawtransaction command. Figure 5 shows the propagation of the transaction $Tr(B)$, the result is a new identifier for the created transaction.



```
sendrawtransaction
02000000000101d6ad88e858dc8a58b48cb5ee1524066ba38bb93494ed58b03551152bf08d4c890
0000000171600145a9555c890f6e0d5753a90c2903ede70e45757c7ffffffff0228230000000000
0017a91431dc48f21516212b42d8c9218f7d321b97948fb8872c0100000000000017a914027f392
11ddd0e234908d0cedb85fcfa8514019c870247304402204c973e2138dc2847d0f69e2a3d49de94
56993a9111cc0e97e2d825a8c4b5e7460220269cd0b8c08b84f7109332f717443a776a03f41007e
89094f67dc5364c733be30121024bc6ff8ef128e307b3e3a06acf7f57418df4c1555ffa11110084
7a12ff70e2a000000000
ce18f033e2c0cb0032194ca93979773d53d17528864cdf4d7c05aab9f9b1f21e
```

Fig. 5. Propagation of $Tr(B)$ transaction to the Bitcoin Testnet network.

Bitcoin software does not allow signing two transactions that spend the same currency [3]. Therefore, Alice uses the Helper Node to sign the $Tr(A)$ transaction. In the next section, multiple transactions spending the same currency will be created and propagated according to the Karame model to observe the probability of success of double-spending attacks on fast Bitcoin payments.

5.4 Experiments

To satisfy Karame's requirements, we connect Alice's node directly to Bob's node, which satisfies requirement 1. We assume that Bob delivers the product to Alice once he sees transaction $Tr(B)$ in his Mempool, which satisfies requirement 3. Finally, to observe the probability of confirmation of transaction $Tr(A)$ to the blockchain, we will modify Alice's Helper Node connections and the propagation time of transaction $Tr(A)$ versus transaction $Tr(B)$.

The propagation of transactions $Tr(A)$ and $Tr(B)$ are made with a time difference shown on the x axis of Figure 6. For example, if the time is equal to 1 second, it means that $Tr(A)$ propagated 1 second after $Tr(B)$, and if time is equal to -4 seconds means that $Tr(A)$ propagated 4 seconds before $Tr(B)$. Therefore, when the time difference is 0, it means that the two transactions were propagated at the same time. For every time difference, 10 attacks on Bitcoin Testnet [12], the probability that $Tr(B)$ belongs to the blockchain is equal to $Tr(A) - 1$.

In the first set of attacks, Bob is connected to eight nodes, including Alice, and the Alice's Help Node connect to eight nodes without relationship to Bob's connections. The experiment aims to observe the probability of success of a double-spending attack in an uncontrolled environment with the Karame's model using the default configuration of the Bitcoin Core Testnet. Figure 6 shows that

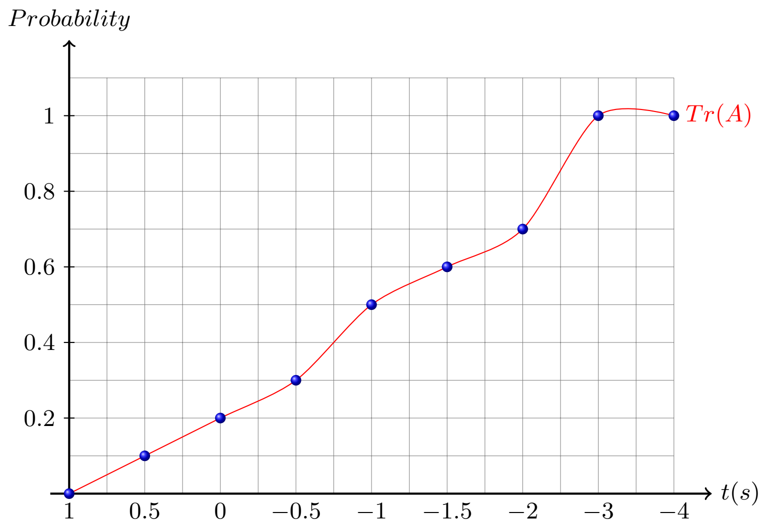


Fig. 6. Probability of success of the malicious transaction, when the attacking node and Bob's node connect to the same number of nodes.

although Bob and Alice's Helper Node have the same eight connections, they are under an uncontrolled environment. Because when the difference in propagation time of transactions is 0, the malicious transaction $Tr(A)$ propagated by Alice's Helper Node has only a 20% chance of confirmation in the Blockchain. Also, when the propagation time of transaction $Tr(B)$ is delayed, the probability of adding transaction $Tr(A)$ to the blockchain increases.

In the second set of attacks, Alice's Helper Node connects to 50 nodes. We emphasize that increasing the number of connections increases the waiting time between the connections of each node. The connection timeout parameter in this experiment is 15 seconds per node. This experiment aims to observe the probability of confirmation of Alice's $Tr(A)$ malicious transaction when connections to Alice's Helper Node are increased. Although the environment is not controlled, the probability of confirmation of the transaction $Tr(A)$ should increase compared to the graph before.

Figure 7 shows a relevant increase in the probability of success of the malicious transaction. For attacks with one second of difference, the confirmation increased by 10% compared to attacks in the previous experiment. Also, the time the malicious transaction $Tr(A)$ reaches 100% of the probability of confirmation is reduced. However, the advantage of transaction $Tr(B)$ continues with a high percentage of success when the two transactions propagate at the same time. This modification of Alice's Helper Node connections shows how easy it is to give the malicious transaction $Tr(A)$ an advantage to satisfy requirement 2.

In the third set of experiments, Alice's Help Node connections increase to 100 nodes. The wait parameter between node connections is the same as in the previous experiment. We find that Orange pi's resources are limited to run this experiment, and a more powerful computer is used (Laptop Core I7 9th Generation with 16 Gb Ram and 1 Gigabit Ethernet Port). The experiment aims to increase the connections of Alice's helper node to 100 nodes and to observe if the probability of confirmation of the malicious transaction $Tr(A)$ increases proportionally. Figure 8 shows the result of increasing the connections of Alice's Help Node, increasing the probability of confirmation of the malicious transaction $Tr(A)$. However, the advantage obtained is low compared to the previous experiment, and the change in the hardware and software is costly.

Discussion Finally, the experiments are an implementation following Karame's model for double-spending attacks successfully and supported by the analysis of the Bitcoin Mempool. We modify the variables to satisfy requirement 2, such as the propagation time difference between the malicious transaction $Tr(A)$ and the transaction trying to pay Bob $Tr(B)$, also the number of connections. We note that there is a high probability of success of the double-spending attacks on fast Bitcoin payments under an uncontrolled environment such as the Bitcoin Testnet. The probability of confirmation $Tr(A)$ can increase with more complex attacks. We highlight that the experiments used hardware with limited resources

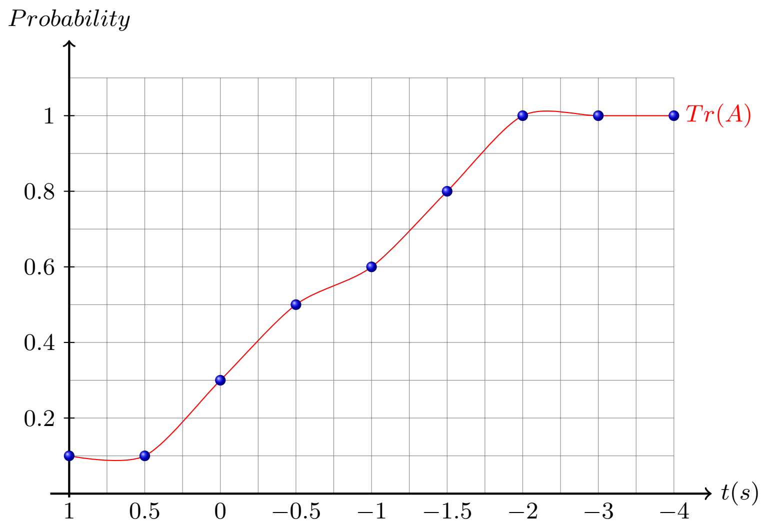


Fig. 7. Probability of success of the malicious transaction, when the attacking node connects to fifty nodes and Bob connects to eight nodes.

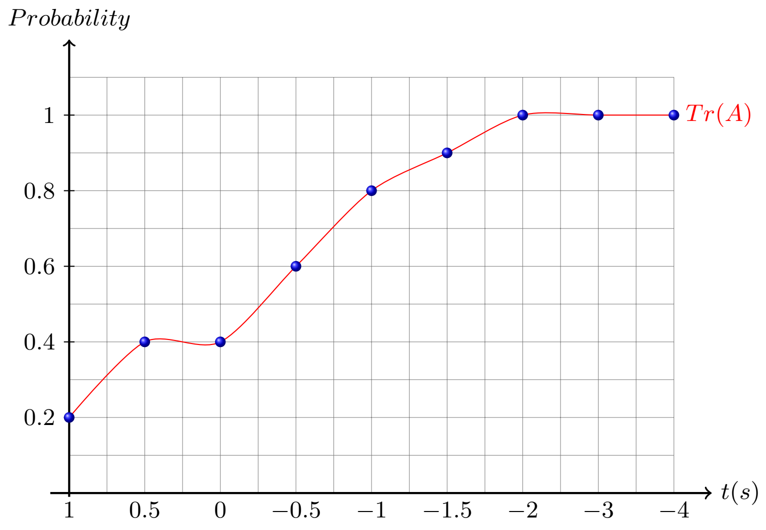


Fig. 8. Probability of success of the malicious transaction, when the attacking node connects to hundreds of nodes and Bob connects to eight nodes.

Orange pi PC, and only necessary to modify the hardware in the last experiment. However, the probability did not increase as expected. Finally, it is possible to increase the probability of success of these attacks if the network connections are analyzed.

6 Conclusions

In this article, Bitcoin's vulnerability to double-spending attacks in fast payment scenarios was shown at a low level. The attacks were implemented on Karame's model and the analysis of transactional processing. The number of connections of Alice's Helper Node and the transactions' propagation time were modified to measure the attack success probability. The experiments showed a 70% success probability using a low-cost device such as Orange pi PC. This vulnerability stops the massive adoption of Bitcoin and leaves an open issue to develop mechanisms that avoid double-spending attacks, as future work remains to analyze the Karame model from a time-based logical distributed view and find the necessary and sufficient requirements for a double spending attack.

References

1. S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system (2009).
2. M. Rosenfeld, Analysis of hashrate-based double spending (2014).
3. S. Nakamoto, Bitcoin web, https://developer.bitcoin.org/ (01-06-2020).
4. F. Quirós, Cajeros automáticos de bitcoin en colombia: ¿dónde están y cuántos hay?, https://es.cointelegraph.com/news/bitcoin-atms-in-colombia-where-are-they-and-how-many-are-there (2020).
5. J. Bastardo, Usé bitcoin para pagar una hamburguesa en burger king y te lo cuento, https://es.cointelegraph.com/news/i-used-bitcoin-to-pay-for-a-burger-at-burger-king (2020).
6. M. Herrmann, Implementation, evaluation and detection of a doublespend-attack on bitcoin (2012).
7. G. O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, S. Čapkun, Misbehavior in bitcoin: A study of double-spending and accountability, ACM Trans. Inf. Syst. Secur. 18 (2015).
8. G. O. Karame, E. Androulaki, S. Capkun, Double-spending fast payments in bitcoin, in: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12, Association for Computing Machinery, New York, NY, USA, 2012, p. 906–917.
9. J. P. Podolanko, J. Ming, Countering double-spend attacks on bitcoin fast-pay transactions, in: ieee-security, 2017, pp. 1–5.
10. T. Bamert, C. Decker, L. Elsen, R. Wattenhofer, S. Welten, Have a snack, pay with bitcoins, in: IEEE P2P 2013 Proceedings, 2013, pp. 1–5. doi:10.1109/P2P.2013.6688717.
11. C. Pérez-Solà, Double-spending prevention for bitcoin zero-confirmation transactions, UCL Discovery (2019).
12. W. Bitcoin, Testnet, https://en.bitcoin.it/wiki/Testnet (18-06-2020).
13. A. M. Antonopoulos, Mastering Bitcoin: Unlocking Digital Crypto-Currencies, 1st Edition, O'Reilly Media, Inc., 2014.

14. Blockchain, Bitcoin developer reference, https://bitcoin.org/en/developer-reference (2020).
15. D. Melo, Github - jdom1824/graph_mempool (2022).
URL https://github.com/jdom1824/Graph_Mempool
16. Blockchain.com, Bitcoin explorer price, url = https://www.blockchain.com/es/explorer (2022).
17. Blockchain.com, Bitcoin explorer mempool size, url = https://www.blockchain.com/es/explorer (2022).
18. S. Xunlong, Orange pi pc, http://www.orangepi.org/orangepipc/ (2015).
19. I. Pečovnik, Armbian, https://www.armbian.com/ (19-06-2020).