A parasite chain attack in IOTA

D. Cai
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands

ABSTRACT

IOTA is a new cryptocurrency that is based on a directed acyclic graph called the *Tangle*. This underlying distributed public ledger technology is fundamentally different from blockchain. IOTA is still in its infancy, which means that not a lot of academic research has been performed yet. One of the most fundamental aspects of a cryptocurrency is that its safety must be guaranteed. The parasite chain attack is one of the potential attacks that is described in the IOTA white paper. In this research paper, the parasite chain attack was used to research some security aspects of IOTA.

Keywords

IOTA, double-spending attack, parasite chain attack, vulnerability.

DISCLAIMER: Little academic research has been performed on IOTA as of yet, since it is a relatively new technology. Therefore I have used non-academic sources throughout my research.

1. INTRODUCTION

In the last few years, cryptocurrencies have seen a dramatic rise in popularity. One of the most recently released cryptocurrencies is called IOTA [5]. While almost all currently existing cryptocurrencies are based on blockchain, IOTA instead uses a new distributed ledger technology called the *Tangle* [15]. This idea was first conceptualized in 2014, with IOTA having been publicly released only in July 2016.[17] The fact that it has not existed that long means that not a lot of academic research has been performed on IOTA yet.

2. BACKGROUND

2.1 IOTA

IOTA [5] is one of the most recently launched cryptocurrencies. It was released in 2017 and, as its name implicates, is focused around the Internet of Things (IoT). IOTA differs from traditional cryptocurrencies by using a different underlying ledger technology. Instead of using blockchain to store and confirm transactions, IOTA uses a Directed Acyclic Graph (DAG) called the Tangle[15]. Transactions are not stored in blocks, but are connected to the Tangle. Furthermore, transactions are not confirmed by miners but they are instead verified by other nodes when they make a transaction themselves. This means that in IOTA there is no distinction between miners and regular nodes.

2.1.1 Weights and approval

In IOTA, transactions have a property called their own weight. The own weight of a transaction is proportional to the amount of PoW that a node has performed [15]. Transactions with higher weights are considered to be more trustworthy than transactions with smaller weights. The own weights contribute to the *cumulative weight* of a transaction, which consists of the transaction's own weight plus the own weights of all transactions that directly or indirectly approve that transaction. It is assumed in the white paper that no entity can create a large amount of transactions with acceptable weights in a short amount of time.

If a transaction is referenced (approved) by another transaction, then the cumulative weight will increase by the cumulative weight of that other transaction. The higher the cumulative weight of a transaction, the more likely it is that it will be approved by the rest of the network.

2.1.2 Transactions and tip selection

Every transaction that is sent to the network has to approve two other transactions, called a branch transaction and a trunk transaction [14]. These transactions are called tips. Tips are transactions that have not been approved by any other transactions yet.

2.1.3 Coordinator and consensus

Consensus is currently controlled by the Coordinator. The Coordinator is a specialized piece of software that is used to periodically issue a normal signed transaction, called a *milestone* [12]. A transaction is considered to be confirmed if and only if it is referenced, either directly or indirectly, by a milestone. The coordinator exists to prevent double-spending attacks, since the network currently does not contain enough hashing power to be intrinsically secure.

The IOTA Foundation plans to remove the Coordinator in the near future [12]. There are several reasons for this. In theory, it allow the Foundation to prioritize transactions and freeze funds. Furthermore, it is a single point of attack and it limits the scalability of the network.

3. PROBLEM STATEMENT

3.1 Double-spending attack

A double-spending attack is an attack in which a malicious user attempts to spend the same digital currency multi-

30^{th} *Twente Student Conference on IT* Feb. 1^{st}, 2019, Enschede, The Netherlands.

ple times [1]. This is a problem that also affects Bitcoin and other cryptocurrencies [3]. It is of paramount importance that the probability of a successful double spend in a cryptocurrency is almost zero. Otherwise the validity of transactions cannot be guaranteed, which will make the cryptocurrency worthless.

There are multiple ways to attempt a double-spending attack in IOTA. A few of them are described in the white paper [15]. These are the large weight attack, the splitting attack and the parasite chain attack. In this research the focus was on the parasite chain attack. It can be performed in the following way:

1. Issue a normal transaction and wait for it to be confirmed.
2. Issue a second double spending transaction that uses the same funds as the normal transaction.
3. Issue many small transactions that directly or indirectly approve the double spending transaction to rapidly increase its cumulative weight.
4. If the cumulative weight of the double-spending transaction outpaces the cumulative weight of the valid transaction, then hopefully the other nodes will build further on the double spending transaction and orphan the normal one.

As explained in section 2.1.3, IOTA has currently implemented a Coordinator to approve transactions, which protects against double-spending attacks. However, they will remove this in the near future and replace it with an algorithm called Markov Chain Monte Carlo (MCMC) [4].

3.2 Research questions

The following questions are central in this research:

1. Is it possible to spam transactions that reference a single transaction without being kicked out of the network?
2. Is it possible to outpace the cumulative weight of a normal transaction by artificially increasing the cumulative weight of a double-spending transaction?
3. Is IOTA centralized?

4. RELATED WORK

IOTA was only released in 2017 and thus not a lot of research has been performed on it as of yet. On the official website of IOTA some academic papers can be found that mostly research the Tangle [6]. In the white paper of IOTA, the author describes an attacker might attempt to "outpace" the network in order to double-spend in the Tangle [15]. This is done by either having multiple transactions all verifying the double-spending transaction, or by creating a double-spending transaction with a very large own weight to approve transactions prior to the legitimate transaction.

5. METHODS AND TOOLS

5.1 Tools

Multiple IOTA API libraries exist for connecting to a full node. The source code of the libraries can be found on the public GitHub page of IOTA [10]. Since I am most familiar with the Python programming language, I have chosen to use the IOTA Python API library called *PyOTA* [11] for this project. This library is compatible with Python versions 3.6, 3.5 and 2.7. Installation instructions are available on the GitHub page.

5.1.1 PyOTA

The PyOTA library [11] contains all the necessary code to connect to a full node and execute all currently available core API calls [7]. PyOTA makes use of the *filters* library [16]. This library makes it easy to create complex data validation and processing pipelines, and can also be used to validate complex JSON structures. PyOTA makes use of this through *Command*s, like the PrepareTransferCommand in *prepare_transfer.py* and the SendTrytesCommand in *send_trytes.py*. It validates a request by passing it through filters and sends the request to the IOTA full node. The response from the node is then returned.

The main module in the PyOTA library is called *api.py*. It contains two API classes that can be used to send HTTP requests for communicating with an IOTA full node. These classes are called StrictIota and Iota. StrictIota exposes only the "core" API methods that are described in the IOTA API reference [7]. On the other hand, Iota additionally also implements wrapper methods for common operations, like the *send_transfer()* method.

The source code of the library has been edited in order to remove certain checks in the code and to be able to send custom transactions to the Tangle. Proof of work is performed remotely on the full node, by calling the API method *attachToTangle* [8].

5.2 Modifications to PyOTA

A description of the modified modules is given below.

5.2.1 iota/api.py

In the *api.py* module only the method *send_transfer()* has been edited, by adding two optional parameters: *branch_transaction* and *trunk_transaction*. This allows the user to specify which transactions they want to reference, so they can reference their own transaction. The method calls a class instance of SendTransferCommand, which is defined in *send_transfer.py*.

5.2.2 iota/commands/extended/prepare_transfer.py

This module contains the PrepareTransferCommand class. As the name implicates, it is used to prepare a transfer by assembling the bundle that should be attached to the Tangle. Upon completion it returns the assembled bundle as tryte strings.

When the optional parameter *inputs* is fulfilled, the passed addresses will be used as input addresses. In the normal unaltered code, it is checked whether the input addresses have sufficient balance to be able to fund the transaction. When trying to double spend this is of course unwanted behavior. This check has been disabled such that the inputs will be used regardless of whether they have a sufficient balance.

5.2.3 iota/commands/extended/send_transfer.py

The SendTransferCommand class combines a PrepareTransferCommand call with a SendTrytesCommand call. The function parameters *branch_transaction* and *trunk_transaction* have been added, which are passed along in the class instance call to SendTrytesCommand.

5.2.4 iota/commands/extended/send_trytes.py

SendTrytesCommand attaches transaction trytes to the Tangle, sends a command to the full node to broadcast

the transaction to neighboring nodes and then sends a command to store them in the local storage of the full node. The GetTransactionsToApproveCommand is called to request a branch transaction and a trunk transaction from the full node through the API, which will be approved by the transaction which is being sent.

The optional request parameters *branchTransaction* and *trunkTransaction* have been added. When these parameters are fulfilled then they will be used as branch- and trunk transaction instead of the transactions that were requested from the node. This makes it possible to choose which transactions will be approved, including ones own transactions.

5.3 Implementation

The modified PyOTA library was used to implement a parasite chain attack. This implementation is based on the algorithm that was described in section 3.1. The parasite chain implementation can be used to spam transactions, and to (in)directly reference a single transaction many time. All non-trivial implementation details are described below.

One of the first obstacles that had to be overcome was allowing for the branch and trunk transactions to be specified when sending a transfer, which allows one to validate ones own transactions. Several PyOTA modules had to be modified to make this possible, the modifications have been described in the previous section.

The second transaction that is sent to the Tangle is a double-spending transaction. This is to ensure that most likely no other transactions will reference the second transaction. To accomplish this, the inputs that were used in the normal transaction are reused in the double-spending transaction. When passing an Address instance to use as input address, it is important to also fill in the optional parameters *key_index*, *security_level*, and *balance*. Without those, the request will not pass the RequestFilter, because the request would be incomplete. It is also important to check whether the address and the specified *key_index* actually belong together, because the API will return a generic internal server error if the wrong *key_index* is provided. This was discovered after many painstaking hours of debugging.

A check also had to be added to check whether the normal transaction has been approved or not. The double-spending transaction has to be sent only after the normal transaction has been approved. The function *get_latest_inclusion()* in *api.py* has been used to check this. This check is performed with 5 second intervals, until it passes.

Lastly, there was an issue with the double-spending transaction indirectly referencing the normal transaction. This meant that the parasite chain would not only increase the cumulative weight of the double-spending transaction, but also that of the normal transaction. This was solved by requesting two transactions to approve from the API before the normal transaction is sent to the Tangle. These two transactions will then be approved by the double-spending transaction. Since they were requested before the normal transaction was sent to the Tangle, it is certain that they do not (in)directly reference the normal transaction.

5.4 Experiments

All experiments have been performed on the Developer Network (Devnet) [9] of IOTA. The Minimum Weight Magnitude (MWM) was set to the Testnet value of 9, instead of the usual Mainnet value of 14 [13]. The reason for this is that the other nodes on the network will generally also set their MWM to 9.

The following experiments have been performed:

5.4.1 Spam approval experiment

In this experiment many zero-value transactions were sent to the network that all (in)directly approved the same transaction. This was done to see how the node where the transactions were sent to would react. A few possible outcomes might be that the node will break the connection, or not process the transactions after a while. This experiment is related to research question 1. It was performed by calling the method *approve_tx()*.

5.4.2 Parasite chain experiment

This experiment expands upon the previous one. This is an implementation of the parasite chain attack, which is used to attempt to rapidly increase the cumulative weight of a double-spending transaction and thereby outpacing the cumulative weight of a normal transaction. This can then be used to draw conclusions for research questions 2 and 3. The test has been performed by calling the method *execute()*.

IOTA currently still uses a Coordinator to approve transactions instead of using the MCMC algorithm as described in the white paper. An official implementation of the MCMC algorithm for determining the confirmation threshold does not exist yet, so it cannot be tested whether that would approve the double-spending transaction or not. The expectation is that the Coordinator will simply ignore the double-spending transaction, even if it has a higher cumulative weight than the normal transaction, because the two transactions conflict.

6. RESULTS

The results were originally acquired by connecting the adapter to *https://nodes.devnet.iota.org:443/*,
which is the URI provided by the IOTA Foundation [9]. However, on 17-Jan-2019 it was discovered that the SSL certificate of the URI had expired a day prior. This made it impossible to connect to that URI, because the adapter would not allow it. This problem was still not resolved a week later, thus a switch had to be made to a different public nodes provider. The URI of that provider is
https://nodes.devnet.thetangle.org:443/.
On 26-Jan-2019 it was noticed that the SSL certificate of the IOTA Foundation URI had been renewed, so it could be used again.

6.1 Spam approval experiment

The experiment has been performed a total of 10 times, with 5 tests having been performed while connected to the public nodes service of *TheTangle.org* and 5 tests while connected to *IOTA.org*. This was done to see if there was an observable difference between the two services. In each test a total of 100 zero-value transactions was to be sent to the network, with every transaction (in)directly referencing the same transaction.

The results of the experiments using *TheTangle.org* can be found in Figure 1. In each test, the first 30 transactions were processed by the node with an average transfer time of 1.03 seconds. However, after the initial 30 transactions the transfer time increases linearly by about 1 second per transaction. After transaction number 60 the node consistently returned a 429 BadApiResponse, with the accompanying message: "Too many requests, use your own hardware." The connection was broken, which stopped the

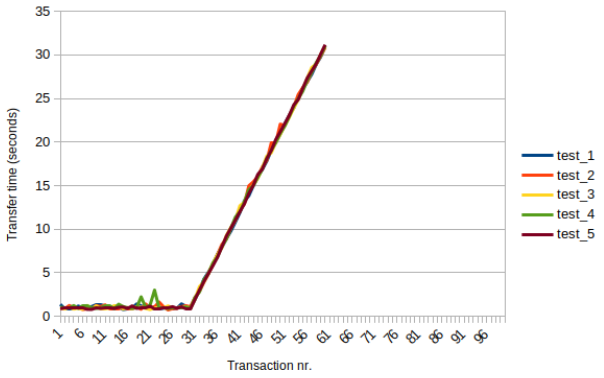

Figure 1. TheTangle.org spam approval results

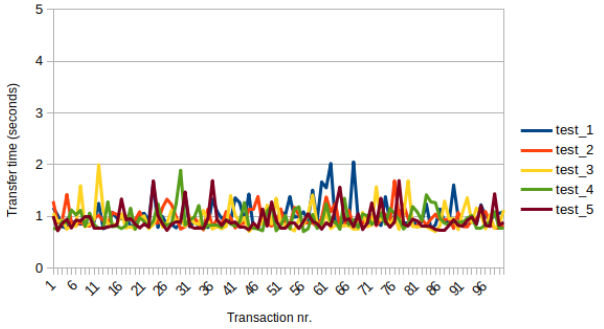

Figure 2. IOTA.org spam approval results

transfer of the remaining 40 transactions. It was, however, still possible to start a new spam approval test right after receiving the BadApiResponse. The behavior also remained the same, with the first 30 transactions being approved in an approximately constant time and the time increasing linearly after that.

The results of the experiments using IOTA.org can be found in Figure 2. It can be clearly seen that the transfer time stays roughly constant for all 100 transactions, which is a different behavior then that of *TheTangle.org*. The five tests been executed immediately after each other, with no observable limitations being enforced by the node. An additional test with 1000 transactions being sent to the network in a continuous loop has also been performed. Again, no decrease in speed or other measures were observed. The transfer of the 1000 transactions took 16 minutes and 24 seconds to complete, which is an average of 0.98 seconds per transfer. This is nearly on par with the average transfer time of 0.96 seconds of the spam test experiments.

6.2 Parasite chain experiment

The aim of this experiment was to see if it is possible to outpace the cumulative weight of a normal transaction, approved by the Tangle, by artificially increasing the cumulative weight of a double-spending transaction with a parasite chain. This test has only been performed using the public nodes service from *TheTangle.org*. The results have been measured by using an online Tangle visualization tool [2] and an online Tangle explorer [18].

A few test runs have been made to test whether the implementation was working correctly. Eventually, three computers were used to perform the final experiment. First, one computer was used to send the initial normal transaction to the Tangle. After waiting for the normal transaction to be confirmed, that computer then sent the second double-spending transaction. Finally, the computer sent parasite transactions that (in)directly approved the double-spending transaction. The transaction hash of the double-spending transaction was copied to the two other computers, so they could also approve that transaction.

The transaction hash of the normal transaction in the final test was

AAMMYQUMSEEWMTPEGDRDBVIZ9ZFEZA9YHMF-JDMQNITFTANJIZN9XT9U9MVSIADTEEPOYZIQUR-BBFXZ999,

and that of the double-spending transaction was

CKVPMXFPIJHUNRQT9PGTDIBCPNZRRONFGMST-DSQWSWLNRXTNVEXRHIFXGETBLLAELFYQSAO-NNEBBNR999.

Using the parasite chain attack it was possible to outpace the cumulative weight of the normal transaction for the entire duration of the experiment, which was 35 minutes. Every time that the full node returned a BadApiResponse, the parasite transactions were simply restarted by calling *approve_tx()* again. However, when looking up the transaction on a Tangle explorer [18], the normal transaction still showed up as confirmed, even though the cumulative weight of the double-spending transaction was consistently higher. At the end of the experiment, the normal transaction had a cumulative weight of 487 and the double-spending transaction had a cumulative weight of 517.

7. DISCUSSION

7.1 Spam approval experiment

A clear difference can be observed between the responses of the *TheTangle.org* public nodes service and *IOTA.org*. *TheTangle.org* has taken measures to protect against the spamming of transactions to the node. This can be seen in Figure 1, where after the first 30 transaction the transfer time increases linearly until transaction 60. At transaction 60 the node simply returns a BadApiResponse, closing the connection. On the other hand, using *IOTA.org* a user is allowed to spam at least 1000 transactions without any noticeable repercussions. This shows that full nodes are free to implement their own measures against spam transactions, but it is not a standard feature. Of course IOTA claims that Proof of Work will protect against spam, but in this case the Proof of Work was not even performed locally on the computer, but instead it was performed remotely by the full node itself. It currently depends on the node that the connection is made to.

It should be mentioned that the results here were acquired with a MWM of 9, which is the Devnet value. On the Mainnet the MWM value is currently 14. This means that the Proof of Work on the Mainnet will most likely take slightly longer than on the Devnet. However, this has no effect on the throughput of transactions.

7.2 Parasite chain experiment

The experiment was performed on the Devnet, which has a MWM of 9. Even though the cumulative weight of the double-spending transaction was higher than that of the normal transaction for an extended period of time, the double-spending transaction was still not confirmed by the Tangle. This is in correspondence with the expectation that the Coordinator would not approve the

double-spending transaction even if the cumulative weight is higher. This also leads to the conclusion that IOTA is currently centralized, since a single entity, the Coordinator, controlled by the IOTA Foundation, gets to decide which transactions are approved and which transactions are not.

As mentioned in section 2.1.3, the IOTA Foundation is planning on removing the Coordinator in the near future. The confirmation of transactions will then be determined by the MCMC algorithm. This will help IOTA become more decentralized. The implementation of the parasite chain attack as described in this paper can then be used to perform further experiments to see whether the algorithm works properly.

In this experiment it was relatively easy to outpace the number of approvals of the normal transaction. The reason for this is that the number of transactions per second on the Devnet were simply not that high at the time of the experiment, only 0.50 TPS. IOTA claims that the MCMC algorithm will be an effective measure against the parasite chain attack, but that claim is based on the assumption that the amount of transactions per second of 'honest' nodes is higher than that of malicious nodes. In future research it might be interesting to consider peaks and valleys in the transaction rate of the Tangle. It might be possible to successfully double-spend using the parasite chain attack in periods of low activity on the Tangle.

8. CONCLUSIONS

IOTA is centralized. This is because the entire system is currently based on transactions being approved by the Coordinator, a central entity controlled by the IOTA Foundation. In the parasite chain experiment, even though the cumulative weight of the double-spending transaction was consistently higher than that of the normal transaction, it was still not confirmed by the rest of the network because the Coordinator had not approved it. The centralized nature of IOTA means that users will have to trust the IOTA Foundation to be benevolent. They have already announced plans to remove the Coordinator in the near future and replace it with a MCMC algorithm. This will make IOTA more decentralized.

The replacement of the Coordinator with the MCMC algorithm means that the confirmation of transactions will be purely based on the general consensus between nodes in the network. The MCMC algorithm is theoretically safe against double-spending attacks, if and only if the transaction rate of honest transactions is higher than that of malicious transactions. In the parasite chain attack experiment it was indeed possible to outpace the cumulative weight of the normal transaction by artificially increasing the cumulative weight of a double-spending transaction using a parasite chain attack. However, this experiment was performed on the Devnet, which at the time of testing had a very low number of transactions per second (0.50). On a more active network this will of course become more difficult.

It was also possible to spam the network with transactions quite easily and seemingly without any repercussions. The *TheTangle.org* public nodes service slowed down the transfer rate after 30 transactions and eventually broke the connection after 60 transactions, but a new connection could be made immediately after. *IOTA.org* appears to have no spam protection measures at all.

Once the Coordinator has been removed, further research can be performed on the MCMC algorithm and how it deals with the parasite chain attacks.

9. REFERENCES

[1] B. Asolo. Double-spending explained. https://www.mycryptopedia.com/double-spending-explained/, November 2018. [Online; accessed 02-Dec-2018].

[2] M. Beck. http://tangle.glumb.de:8080/. [Online; accessed 26-Jan-2019].

[3] Bitcoin.com. What is Bitcoin Double Spending? https://www.bitcoin.com/info/what-is-bitcoin-double-spending, June 2017. [Online; accessed 02-Dec-2018].

[4] A. Gal. Part 3: Cumulative weights and weighted random walks. https://blog.iota.org/the-tangle-an-illustrated-introduction-f359b8b2ec80, February 2018. [Online; accessed 27-Jan-2019].

[5] IOTA Foundation. https://www.iota.org/get-started/what-is-iota. [Online; accessed 19-Nov-2018].

[6] IOTA Foundation. https://www.iota.org/research/academic-papers. [Online; accessed 19-Nov-2018].

[7] IOTA Foundation. API Reference. https://iota.readme.io/v1.5.6/reference. [Online; accessed 20-Jan-2019].

[8] IOTA Foundation. attachToTangle. https://iota.readme.io/reference#attachtotangle. [Online; accessed 27-Jan-2019].

[9] IOTA Foundation. Devnet - Developer Network. https://docs.iota.org/introduction/networks/devnet. [Online; accessed 26-Jan-2019].

[10] IOTA Foundation. IOTA Github. https://github.com/iotaledger. [Online; accessed 20-Jan-2019].

[11] IOTA Foundation. PyOTA. https://github.com/iotaledger/iota.lib.py. [Online; accessed 20-Jan-2019].

[12] IOTA Foundation. Coordinator. Part 1: The Path to Coordicide. https://blog.iota.org/coordinator-part-1-the-path-to-coordicide-ee4148a8db08, November 2018. [Online; accessed 27-Jan-2019].

[13] IOTA Foundation. PoW on the Tangle. https://docs.iota.org/introduction/tangle/proof-of-work, December 2018. [Online; accessed 02-Dec-2018].

[14] L. Lu. In-depth explanation of how IOTA making a transaction (with picture). https://medium.com/@louielu/in-depth-explanation-of-how-iota-making-a-transaction-with-picture-8a638805f905, January 2018. [Online; accessed 27-Jan-2019].

[15] S. Popov. The Tangle. Technical report, IOTA, April 2018.

[16] PyPI. filters 1.3.2. https://pypi.org/project/filters/. [Online; accessed 23-Jan-2019].

[17] D. Sønstebø. IOTA first chapter synopsis. https://blog.iota.org/iota-first-chapter-synopsis-506fdf874437, July 2016. [Online; accessed 20-Jan-2019].

[18] TheTangle.org. Devnet IOTA Tangle Explorer and Statistics. https://devnet.thetangle.org/. [Online, accessed 26-Jan-2019].