The Stability and the Security of the Tangle

Quentin Bramas

The Stability and the Security of the Tangle

Quentin Bramas[1]

[1]ICUBE, University of Strasbourg
bramas@unistra.fr

Abstract

In this paper we study the stability and the security of the Tangle, which is the distributed data structure at the base of the IOTA protocol. The contribution of this paper is twofold. Firstly we present simple model to analyze the Tangle and give the first formal analyzes of the average number of unconfirmed transactions and the average confirmation time of a transaction.

Secondly we define the notion of *assiduous honest majority* that captures the fact that the honest nodes have more hashing power than the adversarial nodes *and* that all this hashing power is constantly used to create transactions. This notion is important because we prove that it is a necessary assumption to protect the Tangle against double-spending attacks, and this is true for any tip selection algorithm (which is a fundamental building blocks of the protocol) that verifies some reasonable assumptions. In particular, the same is true with the Markov Chain Monte Carlo selection tip algorithm currently used in the IOTA protocol.

Our work shows that either all the honest nodes must constantly use all their hashing power to validate the main chain (similarly to the bitcoin protocol) or some kind of authority must be provided to avoid this kind of attack (like in the current version of the IOTA where a coordinator is used).

The work presented here constitute a theoretical analysis and cannot be used to attack the current IOTA implementation. The goal of this paper is to present a formalization of the protocol and, as a starting point, to prove that some assumptions are necessary in order to defend the system again double-spending attacks. We hope that it will be used to improve the current protocol with a more formal approach.

1 Introduction

Since the day Satochi Nakamoto presented the Bitcoin protocol in 2008 [1], the interest in Blockchain technology has grown continuously. More generally, the interest concerns *Distributed Ledger Technology*, which refers to a distributed data storage protocol. Usually it involves a number of nodes (or processes, or agents) in a network that are known to each other or not. Those nodes may not trust each-other so the protocol should ensure that they reach a consensus on the order of the operations that they can perform, in addition to other mechanism like data replication for instance.

The consensus problem has been studied for a long time [2, 3] providing a number of fundamental results. But the solvability of the problem was given in term of proportion of faulty agents over honest agents. But in a trustless network, a adversary can simulate an arbitrary number of nodes in the network. To avoid that, proof systems like Proof of Work (PoW) or (PoS) are used to link the importance of an entity with some external properties (processing power in PoW) or internal properties (the Stake in PoS) instead of simply the number of nodes it controls. The solvability of the consensus is now possible only if less than half of the hashing power is controlled by an adversary (the proportion is reduced to 1/3 if the network is asynchronous).

Bitcoin and the other blockchain technologies are one form of distributed ledger that uses PoW or PoS to elect one node that is responsible for writing data in the blockchain for a given period of time. The "random" selection and the incentive a node may have to execute honestly the protocol

make the whole system secure, as it was shown by several formal analysis [1,4]. Usually, there are properties that hold with high probability i.e., with a probability that tends to one quickly as the time increases. For instance, the order between two transactions do not change with probability that tends to 1 exponentially fast over the time in the Bitcoin protocol, if the nodes executing honestly (or rationally) the protocol have more than the majority of the hashing power.

In this paper we study another distributed ledger protocol called *the Tangle*, presented by Serguei Popov [5] that is used in the IOTA cryptocurrency to store transactions. Very few academic papers exists on this protocol, and there is no previous work that formally analyzes its security.

According to the protocol a PoW has to be done when broadcasting a transaction. This PoW prevents an adversary from spamming the network. However, it is not clear in the definition of the Tangle whether this PoW is also important for its security i.e., avoiding double-spending attack.

In the Tangle, the weight of a set of nodes corresponds to the amount of work they contain. When a new transaction is appended to the Tangle, it references two previous transactions, called tips. The algorithm selecting the two tips is called a *Tip Selection Algorithm*. It is a fundamental parts of the protocol as it is used to reach a consensus. The TSA currently used in the IOTA implementation uses the PoW contained in each transaction to select the two tips.

The main contribution of our paper is to prove that if the TSA depends only on the PoW, then the weight of the honest transactions should exceed the hashing power of the adversary to prevent a double-spending attack. This means that honest nodes should constantly use their hashing power and issue new transactions, otherwise an adversary can attack the protocol even with a small fraction of the total hashing power. Our results is interesting because it is true for any tip selection algorithm i.e., the protocol cannot be secure by simply using a more complex TSA.

The remaining of the paper is organized as follow. Section 2 presents our model and the Tangle. In Section 3 we analyze the average confirmation time and the average number of unconfirmed transactions. In Section 4 we prove our main theorem by presenting a simple double-spending attack. Then, we discuss our results in Section 5 and conclude the paper in Section 6.

2 Model

2.1 The Network

We consider a set $\mathcal{N}$ of processes, called nodes, that are fully connected. Each node can send a message to all the other nodes (the network topology is a complete graph). We will discuss later the case where the topology is sparse.

We assume nodes are activated synchronously. The time is discrete and at each time instant, called round, a node reads the messages sent by the other nodes in the previous round, executes the protocol and, if needed, broadcast a message to all the other nodes. When a node broadcast a message, all the other nodes receive it in the next round. Those assumptions are deliberately strong as they make an attack more difficult to perform.

2.2 The DAG

In this paper, we consider a particular kind of distributed ledger called *the Tangle*, which is a Direct Acyclic Graph (DAG). Each node u stores at a given round r a local DAG G_r^u (or simply G_r or G if the node or the round are clear from the context), where each vertex, called *site*, represents a transaction. Each site has two parents (possibly the same) in the DAG. We say a site *directly confirm* its two parents. All sites that are confirmed by the parents of a site are also said to be confirmed (or indirectly confirmed) by it i.e., there is a path from a site to all the sites it confirms in the DAG (see Figure 1). A site that is not yet confirmed id called a *tip*. There is a unique site called *genesis* that does not have parents and is confirmed by all the other sites.

If the Tangle is used to store the balance of a given currency (like the IOTA cryptocurrency), then a site represents a transaction moving funds from a sender address to a receiver address. Two sites are conflicting if they try to move the same funds to two different receivers i.e., if

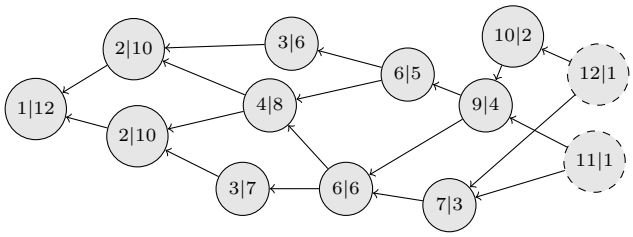

Figure 1: En example of a Tangle where each site has a weight of 1. In each site, the first number is its score and the second is its cumulative weight. The two tips (with dashed border) are not confirmed yet and have cumulative weight of 1.

both executing transactions results in a negative balance for the sender. A site may store several transactions or other kind of messages, but we can assume without loss of generality that it is a single transaction.

At each round, each node may sign one or more transactions. For each transaction, the node selects two parents. The signed transaction becomes a site in the DAG. Then, the node broadcast the site to all the other node.

DAG extension

Definition 1. *Let G be a DAG and A a set of sites. If each site of A has its parents in A or in the tips of G, then we say that A is an extension of G and $G \cup A$ denotes the DAG composed by the union of sites from G and A. We also say that A extends G.*

One can observe that if A extends G, then the tips of G forms a cut of $G \cup A$.

Definition 2. *Let A be a set of sites extending a DAG G. We say A completely extends G (or A is a complete extension of G) if all the tips of $G \cup A$ are in A. In other word, the sites of A confirm all the tips of G.*

The local DAG of a node may contain conflicting sites. If so, the DAG is said to be forked (or conflicting). A conflict-free sub-DAG is a sub-DAG that contains no conflicting sites.

2.3 Tip Selection Algorithm

When signing a transaction and issuing site s, a node u has to select two parents i.e., two previous sites in its own version of the DAG (which appears to be the same as the other nodes at the beginning of the round if nodes are synchronized and the latency if 1). According to the protocol, this is done by executing an algorithm called the *tip selection algorithm* (TSA). The protocol says that the choice of the parents must be done among the sites that have not been confirmed yet i.e., among tips. Also, the two selected parents must not confirm, either directly or indirectly, conflicting sites.

We denote by $\mathcal{T}$ the TSA, which can depend on the implementation of the protocol. For simplicity, we assume all the nodes use the same $\mathcal{T}$ algorithm. As pointed by previous work [5], the TSA is a fundamental factor of the security and the stability of the Tangle.

For our analysis, we assume $\mathcal{T}$ depends only on the topology of the DAG, on the weight of each site in the DAG and on a random source. It is said to be statefull if it also depends on previous output of the TSA by this node, otherwise we say it is stateless.

The algorithm $\mathcal{T}$ is executed by a node when issuing a transaction i.e., when adding a site to the DAG. $\mathcal{T}$ depends on the current version of the DAG and has access to a random source (that is assumed distinct for two different nodes). The random source is used to prevent different nodes to select the same parents when adding a site to the DAG at the same time. When a node has to issue two or more transactions, to avoid selecting the same parent for two consecutive transactions, a node add the first site to its own DAG right after the algorithm $\mathcal{T}$ has computed the parents for

a transactions. Then, the next execution of $\mathcal{T}$ selects the two parents of the other site in the DAG including the first site. However, two distinct nodes cannot cooperate during the same round so two transactions issued by two different nodes may end up having one or two common parents.

Local Main DAG The local DAG of a node u may contain conflicting sites. For consistency, a node u can keep track of a conflict-free sub-DAG $main_u(G)$ that it considers to be its local *main* DAG. If there are two conflicting sites a and $\bar{a}$ in the DAG G, the local main DAG contains at most one of them.

The main DAG of a node is used as a reference for its own view of the world, for instance to calculate the balance associated with each address. Of course this view may change over the time. When new transactions are issued, a node can change its main DAG, updating its view accordingly (exactly like in the bitcoin protocol, when a fork is resolved due to new blocks being mined). When changing its main DAG, a local node may discard a sub-DAG in favor of another sub-DAG. In this case, several sites may be discarded. Of course, we want to avoid such situation or at least ensure that the probability for a site to be discarded tends quickly to zero with time.

The tip selection algorithm decides what are the tips to confirm when adding a new site. Implicitly, this means that the TSA decides what sub-DAG is the main DAG. In more detail, the main DAG of a node at round r is the sub-DAG confirmed by the two sites output by the TSA. Thus, a node can run the TSA just to know what is its main DAG and even if no site has to be included to the DAG.

One can observe that, to reach consensus, the TSA should ensure that the main DAG of all the nodes contain a common prefix of increasing size that represents the transactions everyone agree on.

2.4 Weight and Hashing Power

When a transaction is signed and become a site in the DAG, a small proof of work (PoW) is done. The difficulty of this PoW is called the weight of the site. Initially, this PoW has been added to the protocol to prevent a node to spam a huge number of transactions. In order to issue a site of weight w a processing power (or *hashing power*) proportional to w needs to be consumed. Without loss of generality, we assume that the hashing power denotes the maximum number of sites issued by the nodes.

With the PoW, spamming requires a large amount of processing power, which increases its cost and reduces its utility. For simplicity that the weight of each site is 1.

Then, this notion is also used to compute the *cumulative weight* of a site, which is the amount of work that has been done to deploy this site and all sites that confirm it. Similarly, *the score* of a site is the sum of all weight of sites confirmed by it i.e., the amount of work that has been done to generate the sub-DAG confirmed by it, see Figure 1 for an illustration.

2.5 Adversary Model

Among the nodes, some are honest i.e., they follow the protocol, and some are malicious and behave arbitrarily. For simplicity, we can assume that only one node is malicious and we call this node *the adversary*. The adversary is connected to the network and receive all the transactions like any other honest node. However, it can create (and sign) transactions without broadcasting them, called hidden transaction (or hidden sites).

When an honest node issues a new site, the two sites output by $\mathcal{T}$ must be two tips, at least in the local DAG of the node. Thus, one parent p_1 cannot confirm indirectly the other p_2, because in this case the node is aware of p_2 having a child and is not a tip. Also, a node cannot select a site as parent that it has already picked for a previous site, thus the number of children cannot exceed the number of nodes in the network. This implies the following property.

Property is 1. *In a DAG constructed with a TSA, a site cannot have one parent that confirms the other one. Moreover, the number of children of each site is bounded by the number of nodes in the network.*

The first property should be preserved by an adversary as it is easy for the honest nodes to check and discard a site that does not verify it.

Assiduous Honest Majority Assumption The cumulative weight and the score can be used (among other thing) by a node to select its main DAG. However, even if it is true that a heavy sub-DAG is harder to generate than a light one, there is no relation yet in the protocol between the weight of sites and the hashing power capacity of honest nodes.

We define the *assiduous honest majority assumption* as the fact that the hashing power of honest nodes is constantly used to generate sites and that it is strictly greater than the hashing power of the adversary. In fact, without this assumption, it is not relevant to look at the hashing power of the honest nodes if they do not constantly use it to generates new sites.

Thus, under this assumption, the cumulative weight of the honest DAG grows according to the hashing power of the honest nodes, and the probability that an adversary generates more sites than the honest nodes in a given period of time tends to 0 as the duration of the period tends to infinity. Conversely, without this assumption, an adversary may be able to generates more sites than the honest nodes, even with less available hashing power.

In the Tangle white paper [5] it is mentioned that, if the tip selection algorithm is a simple random tip selection (select two non-conflicting tips randomly), then there should be an assiduous honest majority to prevent an attacker from performing a double-spending attack. However, another tip selection algorithm called Markov Chain Monte Carlo (MCMC) is presented, suggesting that the assiduous honest majority assumption is no more necessary to prevent a double spending attack. To close this discussion, we show in this paper that in a trustless network, and under reasonable assumptions about the protocol, the assiduous honest majority assumption is necessary.

3 Average Number of Tips and Confirmation Time

In this section we study the average number of tips depending on the rate of arrival of new sites. The Tangle white paper [5] contains some errors that have been confirmed in a series of simulation done by [6].

In this section, like in previous analysis [5], we assume that tip selection algorithm is the simple random tip selection that select two tips uniformly at random. Moreover, to be more general, we consider that each site takes h rounds to be received by all the nodes. h can be seen as the latency of the network.

We denote by $N(t)$ the number of tips at time t and $\lambda(t)$ the number of sites issued at time t. One can assume $\lambda(t)$ follow a Poisson distribution of parameter λ. Each new site confirms two tips and we denote by $C(t)$ the number of sites confirmed at time t, among those confirmed sites, $C_{tips}(t)$ represents the number of tips i.e., the number of sites that are confirmed for the first time at time t. Due to the latency, if $h > 1$, a site can be selected again by the TSA. We have:

$$N(t) = N(t-1) + \lambda(t) - C_{tips}(t)$$

Like in the previous work, we assume that the model converges to a stationary state and we look only at the average values of the variables. The convergence might be justified by the fact that if $N(t-1)$ is above average, the tip selection algorithm will likely select more than $\lambda(t)$ tips, hence decreasing the number of tips. Conversely, if $N(t-1)$ is below average, the number of tips increases.

In the stationary state, we can assume for simplicity that exactly λ sites are issued, so that

$$N = N - \lambda + C_{tips} \tag{1}$$

Where N and C_{tips} are respectively the average number of tips and the average number of confirmed tips at a given round. So we want $\lambda = C_{tips}$. Of course the number of selected tips C_{tips} depends on h and on the number of tips in the previous round N.

The following theorem shows the exact like between N, λ and h, and is is used to compute the values of Table 1 that shows several approximate values of N for different values of h.

Theorem 1. *In the stationary state, the average number of tips N is solution to the following equation:*

$$\frac{1}{N^{2\lambda}} \sum_{i=0}^{N-C} \frac{N!}{i!} \left\{ {2\lambda \atop N-i} \right\} = \frac{1}{2} \quad \text{where } C = \lambda + \frac{\lambda^2(h-1)}{N}$$

where h is the latency and λ is the average number of new sites issued every round.

Proof. To better understand the general case, we start with $h = 1$. Then we have $C_{tips} = C$. We want the average number of tips N such that the number C of tips selected by the TSA is λ in average. To solve the equation $C = \lambda$, we view it as a "balls into bins" problem [7] where bins are tips and a ball is thrown in a bin if the tip is selected by the TSA. The number of selected tips corresponds to the number of non-empty bins after throwing 2λ balls into N bins. Let now calculate the probability that the number of non-empty bins is at least λ.

All possible outcomes have the same probability $1/N^{2\lambda}$ to occur. Among all those outcomes, the number of ways one can obtain exactly i empty bins is exactly

$$\frac{N!}{i!} \left\{ {2\lambda \atop N-i} \right\}$$

Where $\left\{ {2\lambda \atop N-i} \right\} = \mathcal{S}_{2\lambda}^{(N-i)}$ is a Stirling number of the second kind representing the number of ways to partition a set of 2λ into $N-i$ non-empty sets. Thus, the probability $P(N, 2\lambda, \lambda)$ to have more than λ non-empty bins is exactly

$$P(N, 2\lambda, \lambda) = \frac{1}{N^{2\lambda}} \sum_{i=0}^{N-\lambda} \frac{N!}{i!} \left\{ {2\lambda \atop N-i} \right\}$$

The stability of the equation (1) occurs when $P(N, 2\lambda, \lambda) = 1/2$, which occurs for N less than 1.258λ. A more complete analysis should be done in order to have an exact value, but our numerical approximation shows that this bound is true for all values of λ (a very little improvement might be done when λ is large). One can observe than if $P(N', 2\lambda, \lambda) \geq 1/2$, then $N \leq N'$, thus the value 1.258λ is an upper bound for N.

When $h > 1$, then $C_{tips} \leq C$. In this case, part of the site selected by the TSA are already selected in past rounds. When stability is reached, $C_{tips} = \lambda$. Among the N visible tips, $\lambda(h-1)$ of them are not tips anymore, so there is a proportion $\frac{N}{N+\lambda(h-1)}$ of sites that are still tips. Since in average the proportion of selected tips compared to selected sites is the same, we have:

$$C_{tips} = \lambda = C \frac{N}{N + \lambda(h-1)}$$

which gives

$$C = \lambda + \frac{\lambda^2(h-1)}{N}$$

By the same analysis as the case $h = 1$ we give the probability $P(N, 2\lambda, C)$ to have more than $C = \lambda + \frac{\lambda^2(h-1)}{N}$ non-empty bins is exactly

$$P(N, 2\lambda, C) = \frac{1}{N^{2\lambda}} \sum_{i=0}^{N-C} \frac{N!}{i!} \left\{ {2\lambda \atop N-i} \right\} \tag{2}$$

The stationary state of equation (1) is obtain for a value N that verifies $P(N, 2\lambda, C) = 1/2$. □

h	1	2	3	4	5	10
N	1.255λ	2.58λ	3.71λ	4.85λ	5.89λ	11.12λ
$Conf$	1.255	2.58	3.71	4.85	5.89	11.12

Table 1: An upper bound on the average number of tips N depending on the latency h and on the number of new sites per round λ. $Conf$ is the Expected number of rounds before the first confirmation. Those values are approximations of N such that $P(N, 2\lambda, C) \geq 1/2$ obtained from equation (2). The approximation may not be true for small value of λ (see figure 3).

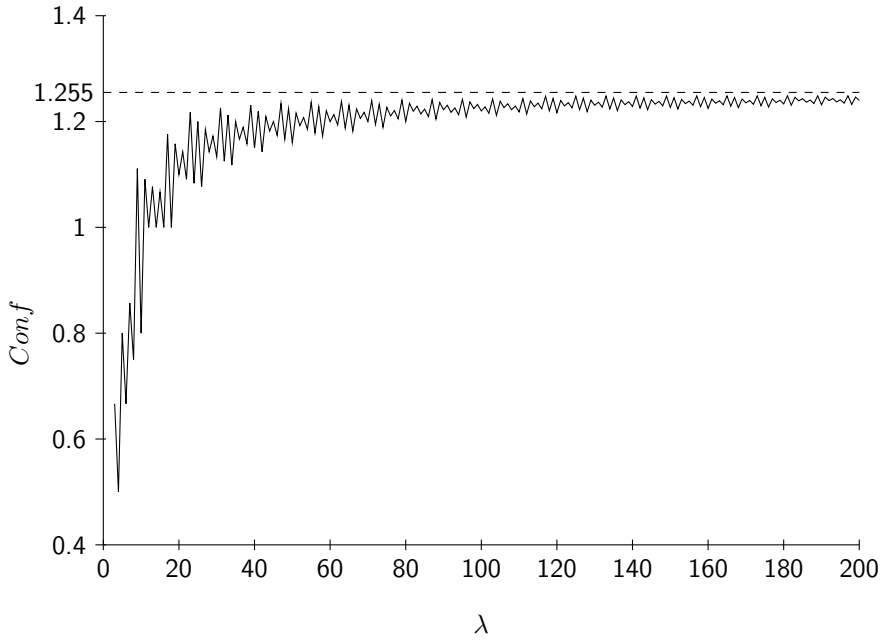

Figure 2: Expected number of round before the first confirmation, depending on the arrival rate of transaction, when $h = 1$. We see that 1.255 is an upper bound. Recall that $\lambda Conf$ refers to the average number of tips at a given time.

Approximate values for such N are given in table 1. The value $N = 1.258\lambda$ when $h = 1$ is consistent with the value found by simulations in previous work [6].

Now that we have the average number of tips, the average confirmation time $Conf$ of a tip is simply given by the fact that, at each round, a proportion λ/N of tips are confirmed. So $Conf = N/\lambda$ rounds are expected before a given tip is confirmed.

4 The Mirror Attack

Our attack is based on a double spending attack i.e., the adversary signs and broadcast a transaction to transfer some funds to a seller to buy a good or a service, and when the seller gives the good (he consider that the transaction is finalized), the adversary broadcast a transaction that conflicts the first one and broadcast other new transactions in order to discard the first transaction. When the first transaction is discarded, the seller is not paid anymore and the funds can be reused by the adversary.

The original motivation for our attack is as follow: after the initial transaction to the seller, the adversary generates the same sites as the honest nodes, forming a sub-DAG with the same topology as the honest sub-DAG (but including the conflicting transaction), hence the name mirror attack. Having no way to tell the difference between the honest sub-DAG and the adversarial sub-

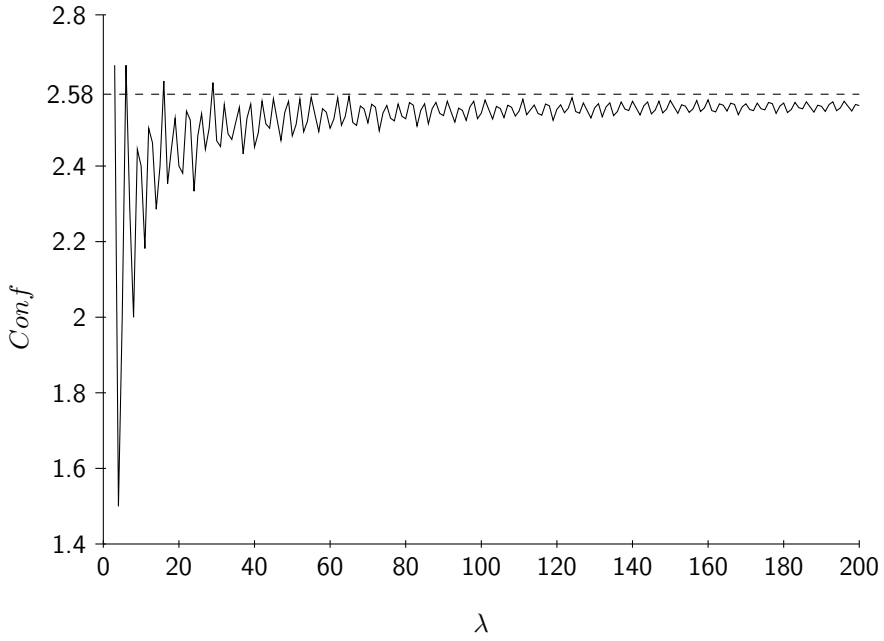

Figure 3: Expected number of round before the first confirmation, depending on the arrival rate of transaction, when $h = 2$. We see that 2.59 is an upper bound when ignoring small values of λ. Recall that $\lambda Conf$ refers to the average number of tips at a given time.

DAG, the latter will be selected by the honest nodes at some point. This approach may not work if the latency of the network is high, because the sub-DAG of the adversary is always shorter than the honest sub-DAG, which is potentially detected by the honest nodes. To counter this, the adversary can generate a sub-DAG that has not exactly the same topology, but that has the best possible topology for the tip selection algorithm. The adversary can then use all its available hashing power to generate this conflicting sub-DAG that will at some point be selected by the honest nodes. We keep the name mirror attack from the original motivation behind it.

For this attack we use the fact that a TSA selects two tips that are likely to be selected by the same algorithm thereafter. For simplicity we captured this with a stronger property: the existence of a maximal deterministic TSA.

Definition 3 (maximal deterministic tip selection algorithm)**.** *A given TSA $\mathcal{T}$ has a maximal deterministic TSA $\mathcal{T}_{det}$ if $\mathcal{T}_{det}$ is a deterministic TSA and for any DAG G, there exists $N_G \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ the following property holds:*

Let A_{det} be the extension of G obtained with $N_G + n$ executions of $\mathcal{T}_{det}$. Let A be an arbitrary extension of G generated with $\mathcal{T}$ of size at most n, conflicting with A_{det}, and let $G' = G \cup A \cup A_{det}$. We have:

$$\mathbb{P}(\mathcal{T}(G') \in A_{det}) \geq 1/2$$

Theorem 2. *Without the assiduous honest majority assumption, and if the TSA has a maximal deterministic tip selection, the adversary can discard one of its transaction that has an arbitrary cumulative weight.*

Proof. Without the assiduous honest majority assumption, we assume that the adversary can generate strictly more sites than the honest nodes. Let W be an arbitrary weight. One can see W has the necessary cumulative weigh a given site should have in order to be considered final. Let G_0 be the common local main DAG of all node at a given round r_0. At this round our adversary can generate two conflicting sites confirming the same pair of parents. One site a is sent to the honest nodes and the other $\bar{a}$ is kept hidden.

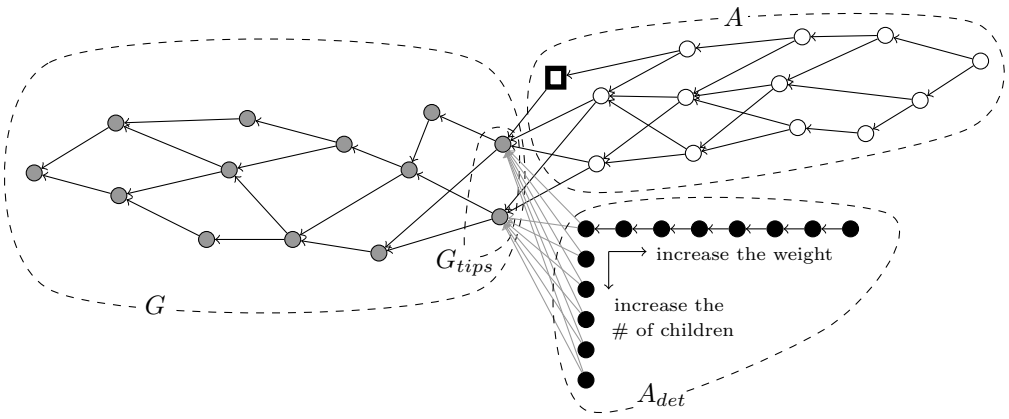

Figure 4: A and A_{det} are two possible extensions of G. The rectangle site conflicts with all site in A_{det} so that when executing the TSA on $G \cup A \cup A_{det}$, tips either from A or from A_{det} are selected. The strategy to construct A_{det} can be either to increase the number of children of G_{tips} or to increase their weight ; both ways are presented here.

The adversary can use $\mathcal{T}_{det}$ the maximal deterministic TSA of $\mathcal{T}$ to generate continuously (using all its hashing power) sites extending $G \cup \{\bar{a}\}$. While doing so, the honest nodes extend $G \cup \{a\}$ using the standard TSA $\mathcal{T}$. After r_W rounds, it can broadcast all the generated sites to the honest nodes. The adversary can choose r_W so that (i) the probability that it has generated N_G more sites than the honest nodes is sufficiently high, and (ii) transaction a has the target cumulative weight W.

After receiving the adversarial extension, by definition 3, honest nodes will extend the adversarial sub-DAG with probability greater than $1/2$. In expectation, half of the honest nodes start to consider the adversarial sub-DAG as their main DAG, thus the honest nodes will naturally converge until they all chose the adversarial sub-DAG as their main DAG, which discard the transaction a.

If the bandwidth of each channel is limited, then the adversary can start broadcasting the sites of its conflicting sub-DAG at round r_W, at a rate two times greater than the honest nodes. This avoids congestion, and at round $r_W + r_W/2$ all the adversarial sub-DAG is successfully received by the honest nodes. Due to this additional latency, the number of sites in the honest sub-DAG might still be greater than the number of sites in the adversarial sub-DAG, so the adversary continues to generate and to broadcast sites extending its conflicting sub-DAG and at round at most $2r_W$, the adversarial extension of G received by the honest nodes has a higher number of sites than the honest extension.

So the same property is true while avoiding the congestion problem. □

Now we have our main theorem, we show that any TSA defined in the Tangle white paper has a corresponding maximal deterministic TSA. To do so we can see that to increase the probability for the adversarial sub-DAG to be selected, the extension of a DAG G obtained by the maximal deterministic TSA should either increase the weight or the number of direct children of the tips of G as shown in Figure 4. We now prove that the three TSA presented in the Tangle white paper [5], (i) the random tip selection, (ii) the MCMC algorithm and (iii) the Logarithmic MCMC algorithm, all have a maximal deterministic TSA, which implies that the assiduous honest majority assumption is necessary when using them (recall that we do not study the sufficiency of this assumption).

4.1 The Random Tip Selection Algorithm

The random tip selection algorithm is the simplest to implement and the easiest to attack. Since it chooses the two tips uniformly at random, an attacker just have to generates more tips than

the honest nodes in order to increase the probability to have one of its tips selected.

Lemma 1. *The Random TSA has a maximal deterministic TSA.*

Proof. For a given DAG G the maximal deterministic $\mathcal{T}_{det}$ always chooses as parents one of the l tips of G. So that, after $n+l$ newly added sites A_{det}, the tips of $G \cup A_{det}$ are exactly A_{det} and no other extension of G of size n can produce more than $n+l$ tips so that the probability that the random TSA select a tip from A_{det} is at least $1/2$. □

Corollary 1. *Without the assiduous honest majority assumption, the Tangle with the Random TSA is susceptible to double-spending attack.*

4.2 The MCMC Algorithm

The MCMC algorithm is more complex than the random TSA. It starts by initially put a fixed number of walker on the local DAG. Each walker performs a random walk towards the tips of the DAG with a probabilistic transition function that depends on the cumulative weight of the site it is located to and its children. In more details, a walker at a site v has a probability $p_{v,u}$ to move to a child u with

$$p_{v,u} = \frac{\exp(-\alpha(w(v) - w(u)))}{\sum_{c \in C_v} \exp(-\alpha(w(v) - w(c))} \tag{3}$$

Where the set C_v is the children of v, and $\alpha > 0$ is a parameter of the algorithm.

The question to answer in order to find the maximal deterministic TSA of MCMC algorithm is: what is the best way to extend a site v to maximize the probability that the MCMC walker chooses our sites instead of another site. The following Lemma shows that the number of children is an important factor. This number depends on the value α.

Lemma 2. *Suppose a MCMC walker is at a site v. There exists a constant C_α such that if v has C_α children of weight n, then, when extending v with an arbitrary set of sites H of size n, the probability that the walker move to H is at most* $1/2$.

Proof. When extending v with n sites, one can choose the number h of direct children, and then how the other sites extends those children. There are several ways to extends those children which changes their weights $w_1, w_2, \ldots, w_h$. The probability p_H for a MCMC walker to move to H is calculated in the following way:

$$\begin{aligned} S_H &= \sum_1^h exp(-\alpha(W - w_i)) \\ \overline{S_H} &= C_\alpha exp(-\alpha(W - n)) \\ S &= S_H + \overline{S_H} \\ p_H &= S_H / S \end{aligned}$$

The greater the weight the greater the probability p_H. Adding more children, might reduce their weights (since H contains only n sites). For a given number of children h, there are several way to extends those children, but we can arrange them so that each weight is at least $n - h + 1$ by forming a chain of length $n - h$ and by connecting the children to the chain with a perfect binary tree. The height l_i of a children i gives it more weight. So that we have $w_i = n - h + l_i$. A property of a perfect binary tree is that $\sum_1^h 2_i^l = 1$. We will show there is a constant C_α such that for any h and any $l_1, \ldots, l_h$, with $\sum_1^h 2_i^l = 1$, we have

$$\overline{S_H} \geq S_H$$
$$C_\alpha \exp(-\alpha(W-n)) \geq \sum_{i=1}^{h} \exp(-\alpha(W-w_i))$$
$$C_\alpha \geq \sum_{i=1}^{h} \exp(-\alpha(h-l_i)) \tag{4}$$

Surprisingly, one can observe that our inequality does not depend on n, so that the same is true when we arrange the sites when extending a site v in order to increase the probability for the walker to select our sites.

Let $f_h : (l_1, \ldots, l_h) \mapsto e^{-\alpha h} \sum_1^h \exp(\alpha l_i)$. So the goal is to find an upper bound for the function f_h that depends only on α.

The function f_h is convex (as a sum of convex functions), so the maximum is on the boundary of the domain, which is either

$$(l_1, \ldots, l_h) = (1, 2, \ldots, h)$$

or

$$(l_1, \ldots, l_h) = (\lceil \log(h) \rceil, \ldots, \lceil \log(h) \rceil, \lfloor \log(h) \rfloor, \ldots, \lfloor \log(h) \rfloor)$$

For simplicity, let assume that h is a power of two so that the second case is just $\forall i, l_i = \log(h)$.

In the first case we have

$$f_h(1, \ldots, h) = \exp(-\alpha h) \frac{\exp(\alpha(h+1)) - \exp(-\alpha)}{\exp(-\alpha) - 1} = \frac{\exp(\alpha) - \exp(-\alpha(h+1))}{\exp(-\alpha) - 1}$$

which tends to 0 when h tends infinity, so it admits a maximum C_α^1

In the second case, we have

$$f_h(1, \ldots, h) = \exp(-\alpha h) h \exp(\alpha \log(h))$$

which again tends to 0 when h tends infinity, so it admits a maximum C_α^2 By choosing $C_\alpha = \max(C_\alpha^1, C_\alpha^2)$ we have the inequality (4) for any value of h.

□

Lemma 3. *The MCMC tip selection has a maximal deterministic TSA.*

Proof. Let G be a conflict-free DAG with tips G_{tips}.

Let T be the number of tips times the number C_α defined in Lemma 2. The T first executions of $\mathcal{T}_{det}$ select a site from G_{tips} (as both parents) until all site from G_{tips} as exactly C_α children.

The next executions of $\mathcal{T}_{det}$ selects two arbitrary tips (different if possible). After T executions, only one tip remains and the newly added sites form a chain.

Let $N_G = 2T$. N_G is a constant that depends only on α and on G. After $N_G + n$ added sites, each site in G_{tips} has a C_α children with weight at least n. Thus, by Lemma 2, a MCMC walker located at a site $v \in G_{tips}$ moves to our extension with probability at least $1/2$. Since this is true for all sites in G_{tips} and G_{tips} is a cut. All MCMC walker will end up in A_{det} with probability at least $1/2$. □

One can argue that this is not optimal and we could have improved the construction of the extension to reduce the number of sites, but we are mostly interested here in the existence of such a construction. Indeed, in practice, the probability for a walker to move to our extension would be higher as the honest sub-DAG A is not arbitrary, but generated with the TSA. Our analyze shows that even in the worst configuration, the adversary can still generate an extension with a good probability of being selected.

Corollary 2. *Without the assiduous honest majority assumption, the Tangle with the MCMC TSA is susceptible to double-spending attack.*

4.3 The Logarithmic MCMC Algorithm

In the Tangle white paper, it is suggested that the MCMC probabilistic transition function can be defined with the function $h \mapsto h^{-\alpha} = \exp(-\alpha \ln(h))$. In more details, a walker at a site v has a probability $p_{v,u}$ to move to a child u with

$$p_{v,u} = \frac{(w(v) - w(u))^{-\alpha}}{\sum_{c \in C_v} (w(v) - w(c))^{-\alpha}} \tag{5}$$

Where the set C_v is the children of v, and $\alpha > 0$ is a parameter of the algorithm. The IOTA implementation currently uses this function with $\alpha = 3$.

With this transition function, the number of children is more important than their weight.

Lemma 4. *The logarithmic MCMC tip selection has a maximal deterministic TSA.*

Proof. Let G be a conflict-free DAG with tips G_{tips}.

Let T be the number of tips.

$\mathcal{T}_{det}$ always selects two sites from G_{tips} in a round-robin manner. After kT executions ($k \in \mathbb{N}$), each site from G_{tips} has $2k$ children.

Let n be the number of sites generated with $\mathcal{T}_{det}$ and A an arbitrary extension of G. Let $v \in G_{tips}$ and C_v be the number of children of v that are in A and $C_{det} = 2n/T$ be the number of children of v generated by $\mathcal{T}_{det}$.

With $w(v)$ the weight of v, we have that $w(v) \leq 2n$ and $C_{det} \leq w(v) - n \leq n$ Let p be the probability that a walker located at v chooses a site generated by $\mathcal{T}_{det}$. We have

$$\begin{aligned} p &\geq \frac{C_{det}(w(v) - 1)^{-\alpha}}{C_v(w(v) - n)^{-\alpha}} \\ &\geq \frac{C_{det}(2n)^{-\alpha}}{C_v(C_{det})^{-\alpha}} = \frac{C_{det}}{C_v T^{\alpha}} = 2n\frac{T^{1-\alpha}}{C_v} \end{aligned}$$

With T a constant and C_v bounded, we have that p tends to infinity as n tends to infinity. This is true for each site of G_{tips}, so after a given number of generated site N_G, the probability that a LMCMC walker located at any site of G_{tips} moves to a site generated by by $\mathcal{T}_{det}$ is greater than $1/2$. □

Corollary 3. *Without the assiduous honest majority assumption, the Tangle with the Logarithmic MCMC TSA is susceptible to double-spending attack.*

5 Discussion

5.1 Sparse Network

The case of sparse networks intuitively gives more power to the adversary as it is more difficult for the honest nodes to coordinate. Let assume that the communication graph is arbitrary. It can be a geometric graph (eg. a grid), a small-world graph with a near constant diameter, or anything else. In order for the protocol to work properly, we assume that for each link between two nodes, there is enough bandwidth to send all the sites generated by the honest nodes and that the usage of each link is a constant fraction of its available capacity. For simplicity we can assume that no conflict occurs from the honest nodes so that, without adversary, the local DAG of each node is its main DAG. Due to multi-hop communications, the local DAGs of the honest nodes may differ but only with respect to the sites generated during the last D rounds, where D is the diameter of the communication graph.

Take an arbitrary node u in this graph. We connect our adversary to node u, so that every sites received by u at round r is received by our adversary at round $r + 1$.

In this arbitrary network, the attack is exactly the same, except for the number of rounds r_W that the adversary waits before revealing the conflicting sub-DAG. Indeed, r_W should be greater to take into account the propagation time and ensure that that the first adversarial site a has a cumulative weight greater than W in all the honest nodes (typically we should wait D more rounds compared to the previous case). As in the previous case, the adversary can broadcast its conflicting sub-DAG while ensuring not to induce congestion, for instance, at a rate two times greater than the honest. The topology of the network does not change the fact that after at most $2r_W + D$ rounds, all the honest nodes have a greater probability to choose the adversarial sub-DAG for their next transactions.

5.2 Trusted Nodes

The use of trusted nodes is what makes IOTA currently safe against this kind of attacks. Indeed, a coordinator node regularly add a site to the DAG, confirming an entire conflict-free sub-DAG. Trusted sites act like milestones and any site confirmed by a trusted site is considered irreversible. However if the trusted node is compromised or is offline for too long, the other nodes are left alone.

The current implementation of IOTA uses a trusted node called *the coordinator* and plans to either remove it, or replace it by a set of distributed trusted nodes.

One can observe that the crypto-currency Byteball [8] uses a special kind of trusted nodes called witnesses. They are also used to resolve conflicts in the DAG.

An important question could be: is the use of trusted node necessary to secure a distributed ledger based on a DAG?

5.3 Avoiding Forks

In the current protocol, conflicting sites cannot be confirmed by the same site. Part of the community already mentioned that this can cause some problem if the latency is high (i.e., if diameter of the communication graph is large). Indeed, by sending two conflicting sites to two different honest nodes, half of the network can start confirming one transaction, and the other half the other transaction. When a node finally receives the two conflicting transactions (assuming that the convergence of is instantaneous) a single site will be assumed correct and become part of all the main DAG of all the honest node. However all the sites confirming the other conflicting site are discarded, resulting in wasted hashing power and increase in confirmation time for those site as a reattachment must occur. The cost for the adversary is small and constant and the wasted hashing power depends on the maximum latency between any two nodes.

On way to avoid this is to include conflicting sites in the DAG (like in the Byteball protocol [8] for instance), by issuing a site confirming the two conflicting sites and containing the information of what site is considered valid and what site is considered invalid. This special site called *decider site* would be the only site allowed to confirm directly or indirectly two conflicting sites. This has the advantage that all the site confirming the invalid one remains valid and do not need to be reattached. However, the same thing can happen if the adversary send two conflicting decider sites to two end of the network. But again, a decider site could be used to resolve any kind of conflict, including this one. Indeed, this may seems like a circular problem that potentially never ends, but every time a decider is issued, a conflict is resolved and the same conflict could have happened even without the decider site. So having decider site should no change the stability of the Tangle and only help avoiding reattaching sites.

6 Conclusion

We presented a model to analyze the Tangle that is more precise than in previous work and we used it to study the average confirmation time and the average number of unconfirmed transaction at any given round.

Then, we defined the notion of assiduous honest majority that captures the fact that the honest nodes have more hashing power than the adversarial nodes *and* that all this hashing power is constantly used to create transactions. We proved that for any tip selection algorithm that has a maximal deterministic tip selection (which is the case for all currently known TSA), the assiduous honest majority assumption is necessary to prevent a double-spending attack on the Tangle.

Our analyze shows that honest nodes cannot stay at rest, should be continuously sign transactions (even empty ones), and use all their hashing power to increase the weight of their local main sub-DAG. If not, their hashing power cannot be used to measure the security of the protocol, like we see for the Bitcoin protocol. Indeed, having a huge number of honest nodes with a very large amount of hashing power cannot prevent an adversary from attacking the Tangle if the honest nodes are not using this hashing power. This conclusion may seem intuitive, but the fact that it is true for all tip selection algorithms (that have a deterministic maximal TSA) is something new that have not been proved before.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980. [Online]. Available: http://doi.acm.org/10.1145/322186.322188

[3] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, 1985.

[4] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 281–310.

[5] S. Popov, "The tangle. white paper," 2016. [Online]. Available: https://iota.org/IOTA_Whitepaper.pdf

[6] B. Kuśmierz, "The first glance at the simulation of the tangle: discrete model," 2016. [Online]. Available: http://iota.org/simulation_tangle-preview.pdf

[7] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press, 2005.

[8] A. Churyumov, "Byteball: a decentralized system for transfer of value," 2016. [Online]. Available: https://byteball.org/Byteball.pdf