An Analysis of Acceptance Policies For Blockchain Transactions

Seb Neumayer[1], Mayank Varia[2], and Ittay Eyal[3]

[1] University of Alaska Anchorage, United States; e-mail sjneumayer@alaska.edu
[2] Boston University, United States; email varia@bu.com
[3] Technion, Haifa, Israel; email ittay@technion.ac.il

Abstract. The standard acceptance policy for a cryptocurrency transaction at most exchanges is to wait until the transaction is placed in the blockchain and followed by a certain number of blocks. However, as noted by Sompolinsky and Zohar [16], the amount of time for blocks to arrive should also be taken into account as it affects the probability of double spending. Specifically, they propose a dynamic policy for transaction acceptance that depends on both the number of confirmations and the amount of time since transaction broadcast.
In this work we study the implications of using such a policy compared with the standard option that ignores block timing information. Using an exact expression for the probability of double spend, via numerical results, we analyze time to transaction acceptance (performance) as well as the time and cost to perform a double spend attack (security). We show that while expected time required for transaction acceptance is improved using a dynamic policy, the time and cost to perform a double spend attack for a particular transaction is reduced.

1 Introduction

Bitcoin and other cryptocurrencies [4, 11] serialize monetary transactions in a data structure called the blockchain to prevent *double spending*: If one transaction spends a certain coin, a subsequent transaction cannot spend that same coin. However, the blockchain occasionally experiences a *re-organization* where a short suffix is replaced. Therefore, a transaction should not be considered as final prematurely, as it would allow an attacker to replace it with another in a *double-spending attack*.

Transactions are typically considered accepted once there is a sufficiently long suffix after them e.g., after five blocks following the block containing the transaction (i.e. 6 confirmations). However, the probability of a successful double-spending attack against a particular blockchain transaction depends on both the number of confirmations and the time elapsed since transaction broadcast.

Consider two cases of a Bitcoin transaction that is accepted as settled after six confirmations. Suppose that in one case, these confirmations take eleven minutes, such as a transaction included in block #485661 [1]. In the other case suppose, these confirmations take nearly three hours, such as a transaction included in block #152217 [2]. The probability of a successful double spend is significantly different between these cases since the attacker has more time to build his chain when the six confirmations take three hours as compared to eleven minutes.

In this paper, we analyze a more general type of acceptance policy first introduced in [16], which takes into account not only the number of confirmations but also the time elapsed since the transaction was broadcast.

The plot shown in Fig. 1 shows the probability of double spend versus the length of time taken for transaction acceptance, assuming transactions are accepted after six confirmations and the attacker has 20% of the network hashrate. Applied to the cases above, when the six confirmations take eleven minutes the probability of double spend is roughly 0.01%, and when the six confirmations take nearly three hours the probability of double spend is roughly 10%. Note how the probability of double spend changes significantly depending on the length of time taken for transaction acceptance and in fact cannot even be bounded below one. This implies accepting a transaction after six confirmations without considering the time taken since transaction broadcast is *insufficient* to ensure the probability of double spend is below a threshold at the time of transaction acceptance. In this paper we analyze security and performance of an acceptance policy that

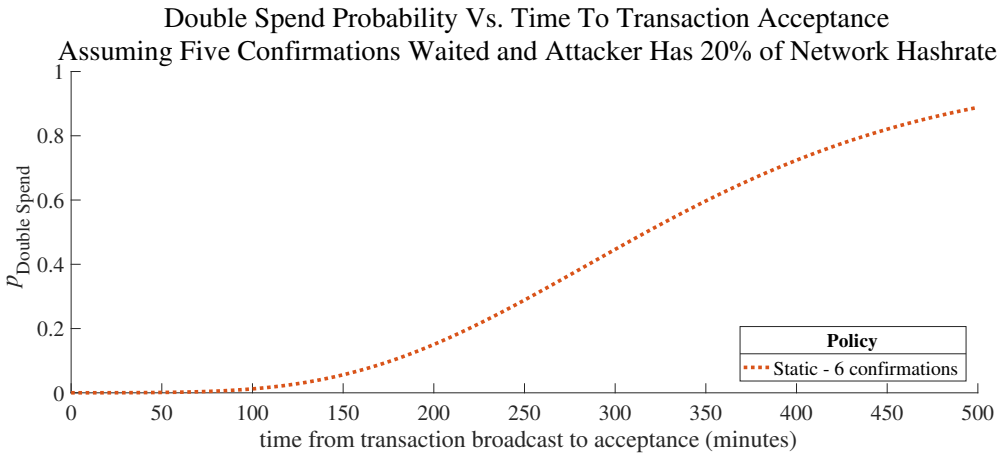

Fig. 1. This plot shows the probability a double spending attack succeeds against a particular transaction versus the amount of time from transaction broadcast to acceptance, assuming the transaction was accepted after six confirmations and the attacker has 20% of the network hashrate. Note that waiting for six confirmations alone is *not sufficient* to ensure a low double spending probability at the time of transaction acceptance.

depends on both confirmations and time since transaction broadcast that bounds the probability of double spend at the time of transaction acceptance.

We call a transaction acceptance policy that waits for a fixed number of confirmations a *static policy*, and call an acceptance policy that depends on both amount of time waited since transaction broadcast and number of confirmations a *dynamic policy*.

Our paper makes the following contributions:

1. A derivation of a closed-form expression for the probability of double spend in terms of a finite sum assuming the attacker persistently attempts to double spend a particular transaction, regardless of cost.
2. An analysis comparing dynamic acceptance policies to static policies, from both the security and performance perspective. Specifically, we analyze dynamic acceptance policies that guarantee the probability of double spend is below a certain threshold, no matter how long the confirmations take to arrive. Via numerical results, we find that dynamic policies reduce the expected waiting time for transaction acceptance as compared to static policies. On the other hand, we find dynamic acceptance policies reduce the attacker's time (and cost) for the successful double spend of a particular transaction as compared to static policies.
3. A discussion of other families of dynamic acceptance policies which may have different security and performance trade-offs.

We overview prior art and proceed by covering each of these contributions in order.

2 Related work

Most previous work has analyzed the probability of double spend solely as a function of the number of confirmations [11, 14]. To the best of our knowledge, Sompolinsky and Zohar [16] were the first to present an acceptance policy based on time since transaction broadcast as well as the number of confirmations. However, they did not analyze of the security and performance properties of dynamic acceptance policies as we do here. Unlike [16], we neglect block propagation delay in our analysis as we make no assumptions on blocksize.

Acceptance policies that depend on the block height that enable different security guarantees are considered in [17], however those ignore the time aspect. Pinzon and Rocha [13] also note that ignoring the time element

does not result in an accurate double-spending probability calculation. [6] considers a 'time-based transaction confirmation verification' policy, however it differs from the dynamic policies considered in this paper.

Blockchains are also exposed to other attacks. The Selfish Mining attack [7] can be combined with other attacks and double-spending [9,12,15]. In the Eclipse attack [10] a node is disconnected from the network by the attacker making it vulnerable to a double spending attack. Considering such combined attacks, a secure acceptance policy should consider timing information.

3 Calculating The Probability of Double Spend

In this section we find the exact closed-form expression for the probability a particular transaction is double spent as a function of the length of time and number of blocks added to the main chain since transaction broadcast. We start by introducing notation and our attacker model. We then derive the probability of double spend in three steps.

We let λ represent the rate at which blocks are added to the main chain when all parties follow the protocol. In Bitcoin, $\lambda = 0.1$ blocks/min and we assume this rate for the remainder of this work. We assume there is one attacker with α fraction of the total network hashrate that is attempting to perform a double spend against a particular transaction. This allows the attacker to generate blocks at a rate of $\alpha\lambda$. We also assume all other miners are honest and produce blocks at a rate of $(1-\alpha)\lambda$. A double spend occurs when the attacker releases a longer blockchain *after* the transaction has been accepted as settled. We also assume the attacker persistently attempts to double spend a particular transaction regardless of time and cost or if the attacker's chain falls many blocks behind.

We now derive the exact probability of double spend expressed as a finite sum in three steps. In the first step, we determine how many blocks ahead the attacker is before the transaction is broadcast. In the next step, we determine how many blocks the attacker is ahead or behind the main chain after the transaction has been accepted as settled. In the third step, we determine the probability that the attacker ever creates a chain longer than the main chain after the transaction has been accepted as settled.

Step 1: To prepare for the double spend attempt, we assume the attacker may mine before the transaction is broadcast. Let p_i denote the probability the attacker is exactly i blocks ahead of the main chain immediately before the transaction is broadcast. We note that if the attacker does not mine on his own chain before the transaction, then $p_0 = 1$. Other distributions over p_i can be chosen depending on the how the attacker is modeled.

Step 2: We now determine the attacker's state after transaction acceptance. We let T be the length of time elapsed from broadcast to transaction acceptance, during which the main chain added N blocks. From the point of view of the receiver of the transaction, the number of blocks the attacker has mined since the transaction broadcast has a Poisson distribution (the receiver does not know the state of the attacker's chain, only the state of the main chain). In the following, we calculate the probability the attacker is a particular number of blocks ahead/behind the main chain immediately after transaction acceptance.

Let p_i^* denote the probability the attacker is exactly i blocks ahead (if attacker is behind, i is negative) the main chain at time T assuming the attacker was ahead by i blocks with probability p_i when the transaction was broadcast. In other words, p_i^* represents the probability the attacker is exactly i blocks ahead/behind the main chain immediately after transaction acceptance. Since the distribution for the number of blocks the attacker finds in time T is Poisson with parameter $\alpha\lambda T$ and the main chain had N confirmations within this time, we have:

$$p_i^* = \sum_{j=-N}^{i} p_{j+N} \frac{e^{-\alpha\lambda T} \left(\alpha\lambda T\right)^{i-j}}{(i-j)!}$$

Step 3: We now determine the explicit form for the probability the attacker creates a chain longer than the main chain that double spends the transaction. If the attacker was ahead of the main chain immediately after transaction acceptance (i.e. the attacker's chain is longer than the main chain), then the attacker can

double spend immediately. Additionally, even if the attacker was not ahead of the main chain immediately after transaction acceptance, there is still a chance the attacker catches up and exceeds the main chain.

The continuous-time Markov process in Fig. 2 models how the attacker may be able to beat the main chain and double spend. Each numbered state represents the number of blocks the attacker is ahead and the labels on the arcs represent transition rates. We note that in addition to the numbered states, there is an absorbing 'Double Spend' state. The probability of double spend, $p_{\text{Double Spend}}$, is the probability of reaching the 'Double Spend' state in this Markov process given some initial distribution (based on p_i^*).

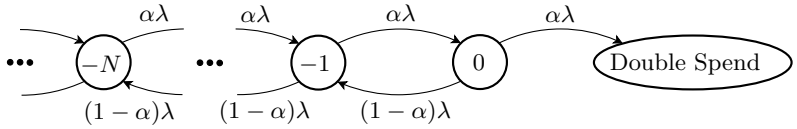

Fig. 2. This continuous-time Markov process models how the attacker may be able to beat the main chain and double spend after transaction acceptance. Each numbered state represents the number of blocks the attacker is ahead and the labels on the arcs represent transition *rates*. We note that in addition to the numbered states there is an absorbing 'Double Spend' state. The probability of double spend is the probability of reaching the 'Double Spend' state in this Markov process.

Given the attacker starts at state i in this Markov process, the probability of reaching the 'Double Spend' state is $\left(\frac{\alpha}{1-\alpha}\right)^{-i+1}$. This is equivalent to the Gambler's Ruin problem discussed in [11] and the result is well-known in random walk theory [8].

Since the probability that the attacker is already capable of double spending at the time of transaction acceptance is $\sum_{i=1}^{\infty} p_i^*$, the probability of a double spend eventually occurring is given by:

$$p_{\text{Double Spend}} = \sum_{i=-N}^{0} p_i^* \left(\frac{\alpha}{1-\alpha}\right)^{-i+1} + \sum_{i=1}^{\infty} p_i^*$$

In order to compute the above probability, we need to need to avoid the infinite summation in the right-most term. Since $\sum_{i=-N}^{\infty} p_i^* = 1$ (immediately after transaction acceptance, the attacker can never be more than N blocks behind), we have:

$$\boxed{p_{\text{Double Spend}} = \sum_{i=-N}^{0} p_i^* \left(\frac{\alpha}{1-\alpha}\right)^{-i+1} + 1 - \sum_{i=-N}^{0} p_i^*}$$

The left sum above refers to the probability of the attacker catching up sometime after the transaction has already been accepted. The remainder of the probability refers to immediately being able to double spend after transaction acceptance.

4 Dynamic Transaction Acceptance Policies

In this section we consider dynamic policies for transaction acceptance that depend on the number of confirmations as well as the amount of time waited since transaction broadcast. Specifically, based upon the amount of time waited, a dynamic policy chooses the minimum number of confirmations required to keep $p_{\text{Double Spend}}$ below a certain threshold. We start by looking at a particular dynamic policy and then use numerical results to analyze properties of comparable static and dynamic policies.

The dynamic policy shown in Fig. 3 ensures the probability of double spend at the time of transaction acceptance is below 1.337%. Note that the number of confirmations required changes with the time elapsed

since transaction broadcast. For example, if the third confirmation arrives ten minutes after transaction broadcast, the transaction is accepted under this policy. However, 60 minutes after transaction broadcast, if the transaction has not yet been accepted then five confirmations would be required for transaction acceptance.

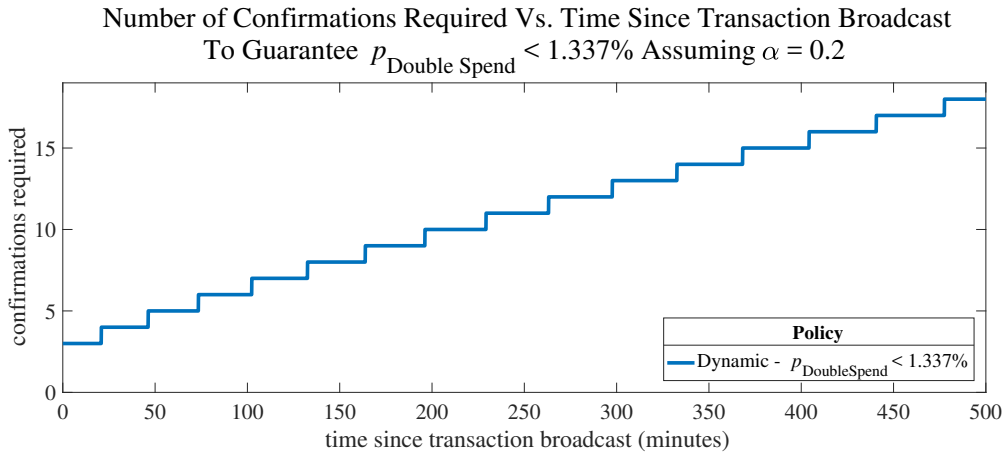

Fig. 3. This plot of a dynamic policy shows the number of confirmations required for a transaction to be accepted as a function of time since transaction broadcast to ensure $p_{\text{Double Spend}} < 1.337\%$. Note that the number of confirmations required changes with the time since transaction broadcast. For example, if the third confirmation arrives ten minutes after transaction broadcast, the transaction is then accepted under this policy. However, if the transaction has not yet been accepted 60 minutes after transaction broadcast, then five confirmations would be required for acceptance.

Fig. 4 shows the double spend probability as a function of time from transaction broadcast to acceptance for the dynamic policy in Fig. 3. Note that the double spend probability at transaction acceptance may be significantly lower than 1.337%. For example, a transaction that is accepted just a few minutes after broadcast has a double spend probability of roughly 0.5%. Comparing to Fig. 1, we see that the dynamic policy bounds the double spend probability at the time of transaction acceptance unlike the static policy.

We note that the parameter space for dynamic acceptance policies allows for continuous policy adjustments as compared to static policies where only the number of confirmations can be adjusted (e.g. wait for 3, 4, or 5 confirmations).

4.1 Comparing Static and Dynamic Acceptance Policies

In the following, we present numerical results based on the exact analysis in Section 3. In the following numerical results, we have assumed $\alpha = 0.2$ unless otherwise noted. We initially find a static and dynamic policy such that their expected double spend probabilities are equal. We then compare the distribution of transaction acceptance time and well as double spend probability for both these policies. We also find the distribution of the required time to perform a double spend attack under both static and dynamic policies. We find that for the same level of security, dynamic policies reduce the expected waiting time for transaction acceptance as compared to static policies. On the other hand, we find dynamic acceptance policies reduce the attacker's time (and cost) for the successful double spend of a particular transaction as compared to static policies. The simulation is written in Matlab and uses the finite-sum expression for the probability of double spend derived in Section 3. All code is open-sourced and can be found at [3].

Comparable static and dynamic policies are shown in Fig. 5. The static policy, shown in dashed red, accepts a transaction after 6 confirmations, regardless of the time since the transaction was broadcast. This results in an $E[p_{\text{Double Spend}}]$ of 0.8754%. The dynamic policy, shown in solid blue, ensures $p_{\text{Double Spend}} < 1.337\%$ at the time of transaction acceptance, which also results in $E[p_{\text{Double Spend}}]$ of 0.8754%. So these policies are

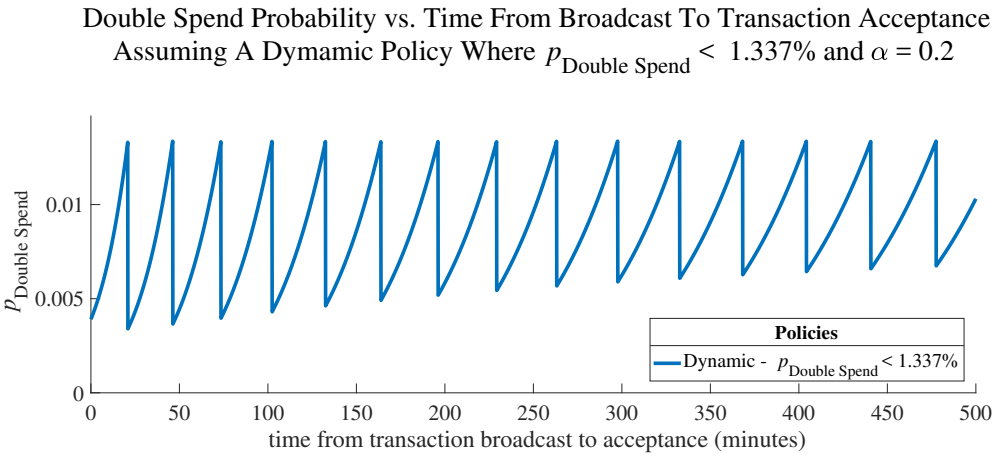

Fig. 4. This plot shows the double spend probability as a function of time from transaction broadcast to acceptance for the dynamic policy in Fig. 3. Note that the double spend probability at transaction acceptance may be significantly lower than 1.337%. For example, a transaction that is accepted just a few minutes after broadcast has a double spend probability of roughly 0.5%. Comparing to Fig. 1, we see that the dynamic policy bounds the double spend probability at the time of transaction acceptance unlike the static policy.

comparable in the sense they result in the same expected probability of double spend when $\alpha = 0.2$. Note the dynamic policy requires fewer confirmations than the static policy up to 70 minutes since transaction broadcast but requires more confirmations after 100 minutes since transaction broadcast.

Fig. 6 shows the distribution of time taken for transaction acceptance for the static and dynamic policies shown in Fig. 5. Note that up to 70 minutes after transaction broadcast, the probability of having accepted a particular transaction is larger with the dynamic policy because fewer confirmations are required. For example, 50 minutes after transaction broadcast, with probability 55% the transaction has already been accepted by the dynamic policy, whereas the transaction has only been accepted with probability 20% under the static policy.

The probability of double spend at the time of transaction acceptance depends on the time from transaction broadcast to acceptance which in turn depends on the timing of block arrivals. Since the time between block arrivals are stochastic, the probability of double spend has a distribution. Fig. 7 shows this distribution of $p_{\text{Double Spend}}$ at the time of transaction acceptance for the static and dynamic policies shown in Fig. 5. The plot essentially combines the distribution of time for transaction acceptance in Fig. 6 with the double spend probability at the time of transaction acceptance in Figs. 1 and 4. The $p_{\text{Double Spend}}$ under the dynamic policy never exceeds 1.337%. This implies $P(p_{\text{Double Spend}} < X) = 1$ for all $X > 1.337\%$ and hence the solid blue CDF curve is 1 for 1.337% and beyond. Compare this to the static policy where the $p_{\text{Double Spend}}$ at the time of transaction acceptance is less than 1.337% with probability 0.8. This implies $P(p_{\text{Double Spend}} < 1.337\%) = 0.8$ and hence the dashed red CDF curve is 0.8 at 1.337%. Also note how the dynamic policy protects against the downside of a large $p_{\text{Double Spend}}$; with probability 0.95, the dynamic policy results in $p_{\text{Double Spend}} < 1.33\%$ whereas the static policy only results in $p_{\text{Double Spend}} < 0.04$.

Fig. 8 shows the expected amount of time to accept a transaction as final versus the 99th percentile of the double spend probability at transaction acceptance, for a spectrum of dynamic and static policies. We consider the 99th percentile of the double spend probability (opposed to the expected double spend probability) in order to capture the 'near worst-case' security effects of the policies. The red dots represent the static policies (e.g. wait for 1, 2, 3, etc. confirmations), and the accompanying red dashed plot shows the lowest expected acceptance time using static policies. Note that for a particular value of the 99th percentile of the double spend probability, the expected amount of time waited using the fixed policy is generally larger than the dynamic policy. Roughly speaking, this means for the same protection against the near worst-case, the dynamic policy requires waiting for less time for transaction acceptance.

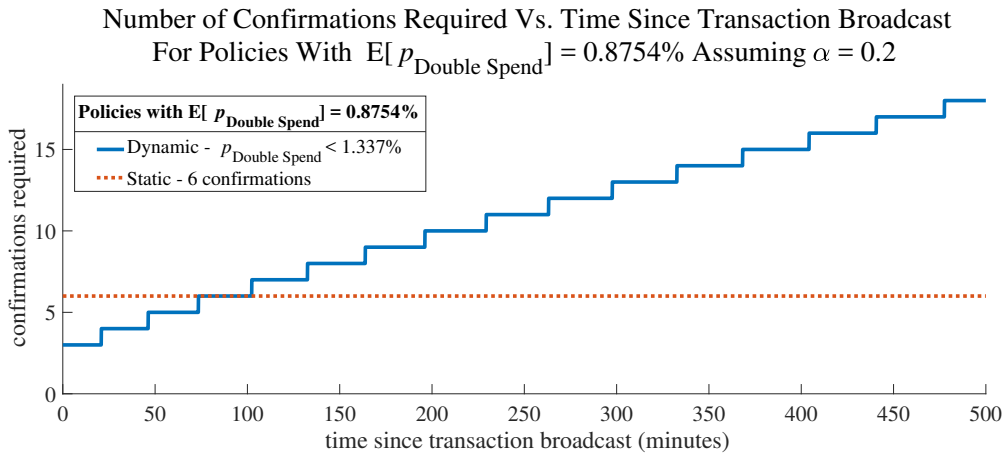

Fig. 5. The plot shows a comparable static and dynamic policy. The static policy, shown in dashed red, accepts a transaction after 6 confirmations, regardless of the time since the transaction was broadcast. This results in an $E[p_{\text{Double Spend}}]$ of 0.8754%. The dynamic policy, shown in solid blue, ensures $p_{\text{Double Spend}} < 1.337\%$ at the time of transaction acceptance, which also results in $E[p_{\text{Double Spend}}]$ of 0.8754%. Note the dynamic policy requires fewer confirmations than the static policy up to 70 minutes since transaction broadcast but requires more confirmations after 100 minutes since transaction broadcast.

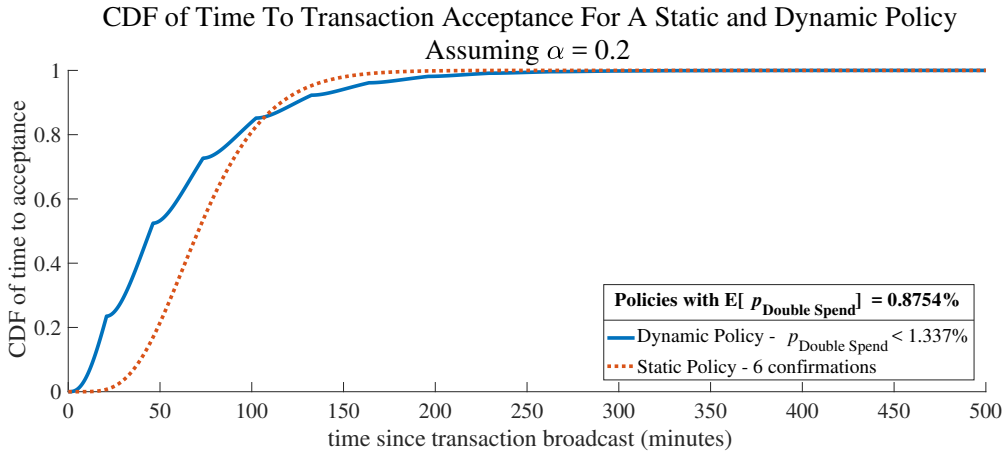

Fig. 6. This plot shows the distribution of time taken for transaction acceptance for the static and dynamic policies shown in Fig. 5. Note that up to 70 minutes after transaction broadcast, the probability of having accepted a particular transaction is larger with the dynamic policy because fewer confirmations are required. For example, 50 minutes after transaction broadcast, with probability 55% the transaction has already been accepted by the dynamic policy, whereas the transaction has only been accepted with probability 20% under the static policy.

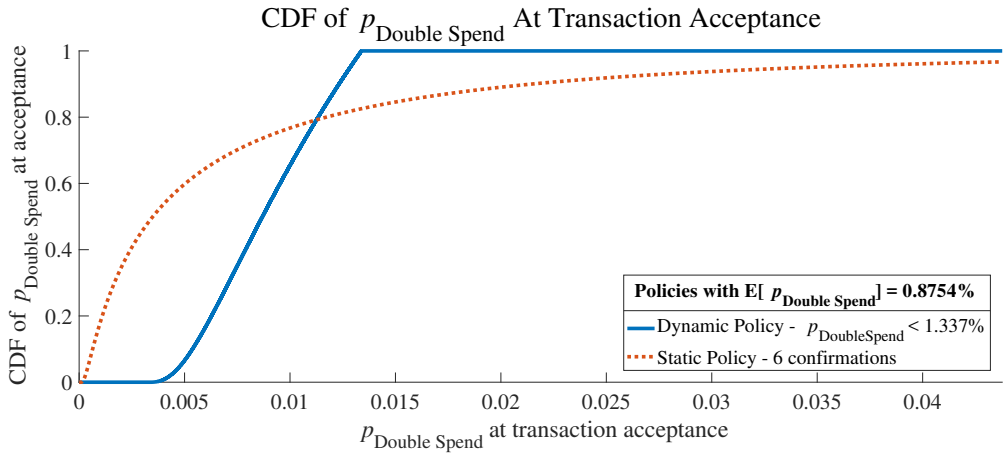

Fig. 7. The plot shows the distribution of $p_{\text{Double Spend}}$ at the time of transaction acceptance for the static and dynamic policies shown in Fig. 5. The plot essentially combines the distribution of time for transaction acceptance in Fig. 6 with the double spend probability at the time of transaction acceptance in Figs. 1 and 4. The $p_{\text{Double Spend}}$ under the dynamic policy never exceeds 1.337%. This implies $P(p_{\text{Double Spend}} < X) = 1$ for all $X > 1.337\%$ and hence the solid blue CDF curve is 1 for 1.337% and beyond. Compare this to the static policy where the $p_{\text{Double Spend}}$ at the time of transaction acceptance is less than 1.337% with probability 0.8. This implies $P(p_{\text{Double Spend}} < 1.337\%) = 0.8$ and hence the dashed red CDF curve is 0.8 at 1.337%. Also note how the dynamic policy protects against the downside of a large $p_{\text{Double Spend}}$; with probability 0.95, the dynamic policy results in $p_{\text{Double Spend}} < 1.33\%$ whereas the static policy only results in $p_{\text{Double Spend}} < 0.04$

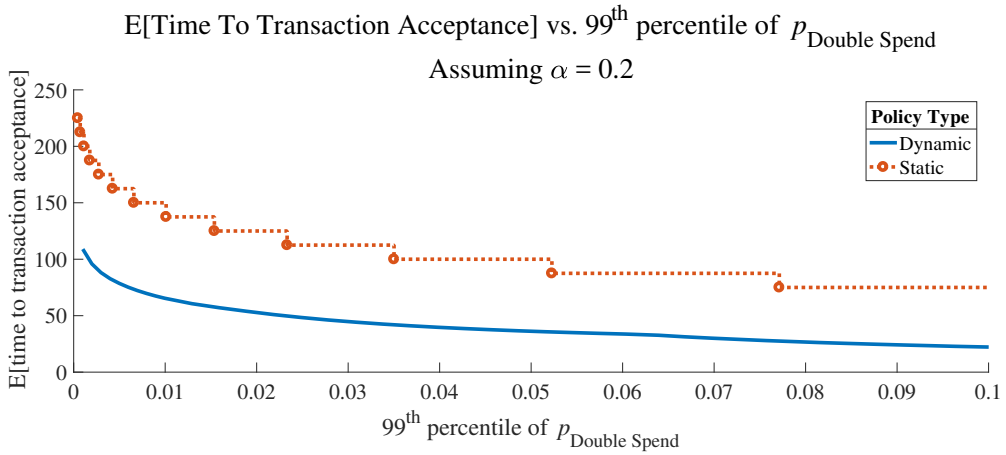

Fig. 8. The plot shows the expected amount of time to accept a transaction as final versus the 99th percentile of the double spend probability at transaction acceptance, for a spectrum of dynamic and static policies. The red dots represent the static policies (e.g. wait for 1, 2, 3, etc. confirmations), and the accompanying red dashed plot shows the lowest expected acceptance time using static policies. Note that for a particular value of the 99th percentile of the double spend probability, the expected amount of time waited using the fixed policy is generally larger than the dynamic policy.

Fig. 9 shows the expected probability that a particular transaction has been double spent versus the time elapsed since the transaction broadcast for the static and dynamic policies shown in Fig. 5. We note the dynamic policy approaches E[$p_{\text{Double Spend}}$] more quickly. So roughly speaking, this dynamic acceptance policy reduces the attacker's burden (in time and cost) to double spend against a particular transaction as compared to the static policy since the transaction will be double spent more quickly with the dynamic policy. This is essentially because if the transaction is accepted shortly after transaction broadcast, then the attacker has to overcome fewer blocks with the dynamic policy as compared to the static policy.

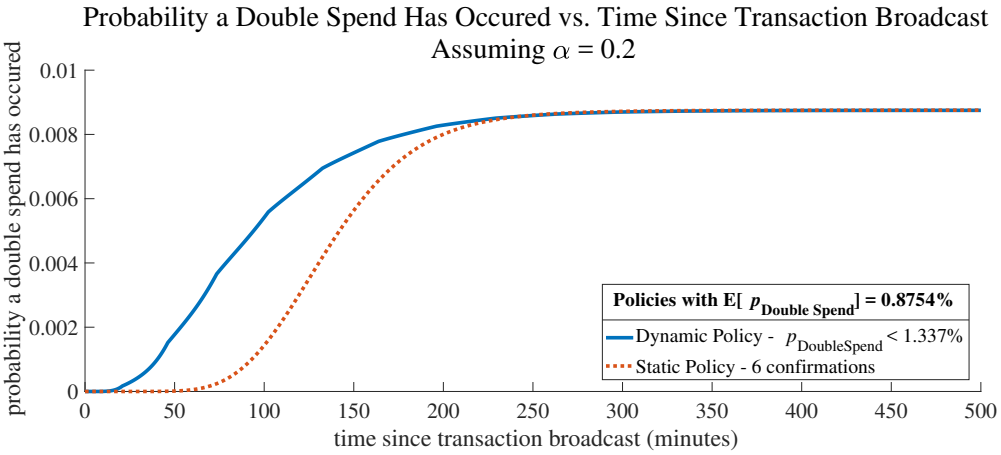

Fig. 9. This plot shows the expected probability that a particular transaction has been double spent versus the time elapsed since the transaction broadcast for the static and dynamic policies shown in Fig. 5. We note the dynamic policy approaches E[$p_{\text{Double Spend}}$] more quickly. So roughly speaking, this dynamic acceptance policy reduces the attacker's burden (in time and cost) to double spend against a particular transaction as compared to the static policy since the transaction will be double spent more quickly with the dynamic policy. This is essentially because if the transaction is accepted shortly after transaction broadcast, then the attacker has to overcome fewer blocks with the dynamic policy as compared to the static policy.

The the 99th percentile of the probability of double spend at the time of transaction acceptance versus α is shown in Fig. 10 for the static and dynamic policies in Fig. 5. Recall, α is the fraction of total network hashrate belonging to the attacker. Roughly speaking, this plot shows for a particular transaction the dynamic policy reduces the near worst-case risk (99th percentile of the double spend probability) over a wide range of attacker hashpower.

5 Other Families of Dynamic Policies and Future Work

In this paper we analyzed a transaction acceptance policy that ensured the double spend probability is below a certain threshold at the time of transaction acceptance. In the future, the same analysis can be done under a different network and threat models that may have different security/performance tradeoffs. For example, acceptance policies can be analyzed assuming the presence of block delays as in [16]. Alternatively, consider the case when mining incentive comes mostly from transaction fees instead of a fixed block reward. This assumption changes the distribution of time between blocks and so the double spending probability and analysis of transaction policies would change [5].

By relaxing the constraint on the maximum probability of double spend, it is possible to construct other families of dynamic transaction acceptance policies. For example, consider the policy that minimizes the expected waiting time for transaction acceptance over all policies with a particular expected probability of double spend. Alternatively, consider the policy that maximizes the amount of time (and hence cost) taken for the attacker to double spend over all policies with a particular expected probability of double spend.

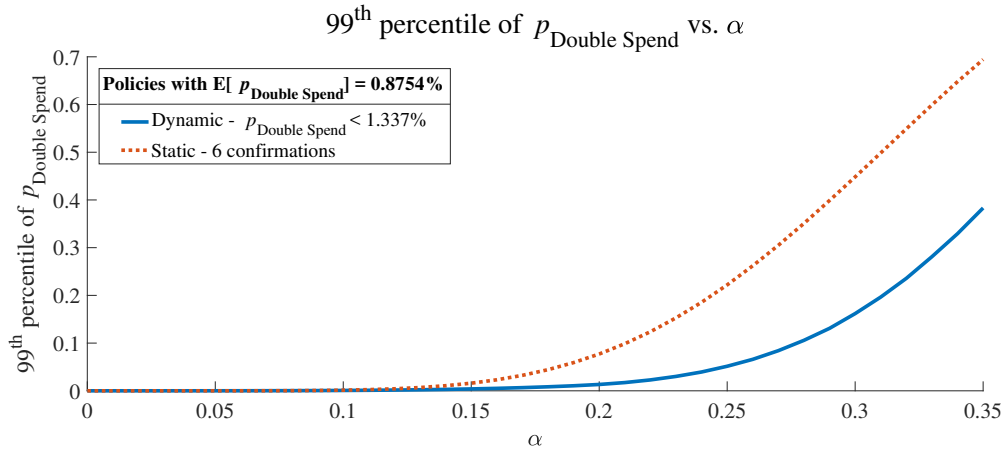

Fig. 10. This plot shows the the 99th percentile of the probability of double spend at the time of transaction acceptance versus α for the static and dynamic policies shown in Fig. 5. Recall, α is the fraction of total network hashrate belonging to the attacker. Roughly speaking, this plot shows for a particular transaction the dynamic policy reduces the near worst-case risk (99th percentile of the double spend probability) over a wide range of attacker hashpower.

These families of acceptance policies may have significantly different performance and security trade-offs compared to the policies considered here.

In addition to determining the probability of double spending a particular transaction, it is possible for other future work to consider the effects acceptance policies on the probability of successfully double spending *any* transaction over a *window* of time (not just a single transaction).

References

1. https://blockchain.info/block-height/485661
2. https://blockchain.info/block-height/152217
3. https://github.com/SebbySebbyBinx/acceptancePolicies
4. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In: Security and Privacy (SP), 2015 IEEE Symposium on. pp. 104–121. IEEE (2015)
5. Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A.: On the instability of bitcoin without the block reward. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 154–167. ACM (2016)
6. Dmitrienko, A., Noack, D., Yung, M.: Secure wallet-assisted offline bitcoin payments with double-spender revocation. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. pp. 520–531. ACM (2017)
7. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: International conference on financial cryptography and data security. pp. 436–454. Springer (2014)
8. Gallager, R.G.: Stochastic processes: theory for applications. Cambridge University Press (2013)
9. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 3–16. ACM (2016)
10. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on bitcoin's peer-to-peer network. In: USENIX Security Symposium. pp. 129–144 (2015)
11. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
12. Nayak, K., Kumar, S., Miller, A., Shi, E.: Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In: Security and Privacy (EuroS&P), 2016 IEEE European Symposium on. pp. 305–320. IEEE (2016)
13. Pinzón, C., Rocha, C.: Double-spend attack models with time advantange for bitcoin. Electronic Notes in Theoretical Computer Science 329, 79–103 (2016)

14. Rosenfeld, M.: Analysis of hashrate-based double spending. CoRR abs/1402.2009 (2014), http://arxiv.org/abs/1402.2009
15. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 515–532. Springer (2016)
16. Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in bitcoin. Financial Cryptography and Data Security (2015)
17. Sompolinsky, Y., Zohar, A.: Bitcoin's security model revisited. arXiv preprint arXiv:1605.09193 (2016)