

Optimal Attack Reward under Saito Networks

Abstract

[The Bitcoin Whitepaper](#) introduced the idea of an independent *self sustaining* network. It is commonly understood that Bitcoin, and all Proof of Work networks are vulnerable to '51%' attacks. In this paper we outline how Saito class networks are not only unvulnerable to 51% attacks, but make all such attacks unprofitable for attackers.

Assumptions

Methods

Terms

x = expected return of the attack
f = all transactions fees in the block
p = proportion of attacker fees in blocks
d = depth in number of blocks attacker waits to mine
c = cost of mining a golden ticket
Ps = paysplit (division of block reward between router and miner)
Pw = powsplit (division of pay between minter and stakers)

Calculating Attack Reward

Saito Basics

In Saito the Block Reward is made up of transaction fees and a small injection of tokens from the treasury. Here we will concentrate on block rewards consisting entirely of transaction fees.

In Saito the block reward does not go to the block creator. Rather, the reward is split between the routing network and a miner. The miner creates the randomness to select the node from the routing network that wins the routing reward. The miner is rewarded for providing this randomness with a prize.

The proportion of the block reward awarded to the routing network and miner is known as the 'paysplit'. Here we will concentrate on Saito's standard implementation of a paysplit of 0.5.

The randomness is created by the miner solving a cryptographic puzzle on the hash of the proceeding block. The solution to the puzzle is called a 'golden ticket'. If no golden ticket is found, no block reward is paid to the network and no prize is paid to the miner.

In early versions of Saito unallocated block reward was moved into the network treasury. Here we analyse a refined version in which token holders can stake their tokens. Unpaid routing and mining rewards are then paid to stakers.

This represents an improvement to the Saito protocol, as it prevents attackers from recycling these rewards into the treasury, then recovering them as later block reward.

The portion of revenue that goes to stakers can be determined using mining difficulty. Specifically mining difficulty can adjust to ensure an average of one golden ticket per n blocks.

Description of the attack and parameters

We are assuming that an attacker is in possession of enough hashpower to control difficulty and able to mine at approaching 100% of total hash on the network. For example a small new Saito class network with few nodes CPU mining in Javascript, and an attacker with a spare Bitcoin mining farm.

It is also assumed that the attacker has sufficient tokens and network connectivity to fill blocks with attack transactions routed only to themselves.

Given the attacker's ability to control difficulty we are assuming they will be completely successful in mining golden ticket solutions when they choose. Though exponential increases in mining difficulty will ensure a set ratio of golden tickets per block.

Given these conditions the expected reward for an attack over time on the network will be:

$$\text{minig reward} + \text{routing reward} - \text{mining costs}$$

Mining Reward

We are presuming the miner uses one of the golden ticket solutions found so their expected reward includes the miner reward or golden ticket prize.

$$f \cdot P_s \cdot n$$

A miner cannot get paid for multiple golden ticket solutions, so for n Unknown character1 the expected reward is simply:

$$f \cdot P_s$$

Routing Payment

If the attacker finds a golden ticket that pays their own node they are paid the entire block reward. The expected reward for a block into which the attacker has stuffed p of their own fees is:

$$f \cdot P_s \cdot p$$

But, the attacker is only getting the honest fees back, so for a single golden ticket solution the payout is the portion of fees that go to the routing reward $f \cdot P_s$ minus the fees the attacker put into the block $f \cdot p$:

$$f \cdot P_s \cdot p - f \cdot p$$

Hashing to find n solutions yields an expected outcome of:

$$f \cdot P_s \cdot (1 - (1 - p)^n) - f \cdot p$$

Given also that the attacker is waiting d blocks to hash, to optimise the number of blocks paying out in their favor, the expected routing reward becomes:

$$d \cdot (f \cdot P_s \cdot (1 - (1 - p)^n) - f \cdot p)$$

Mining Cost

Given the attacker's access to hashpower mining cost is a simple function. The expected cost of mining n golden tickets is: $c \cdot n$. But the attacker will stop mining if they have found a solution that has solved all outstanding blocks.

The chance that a solution solves all blocks in the attacker's favour is p^d . So the expected saving per golden ticket, after the first is $(n-1) \cdot p^d$. The cost of mining when $n > 1$ is then:

$$c \cdot n - ((n-1) \cdot p^d)$$

Total expected attack reward

Combining the above we have:

For n Unknown character1:

$$f \cdot Ps \cdot nd \cdot (f \cdot Ps \cdot (1 - (1-p)^n) - f \cdot p) - c \cdot n$$

For n Unknown character = 1:

$$f \cdot Ps \cdot nd \cdot (f \cdot Ps \cdot (1 - (1-p)^n) - f \cdot p) - (c \cdot n - ((n-1) \cdot p^d))$$

Results

Example 1.

Parameter	Value
Honest Fees	100
Attacker Fees	100
GT Reward	100
Cost/GT	50
PaySplit	0.5
PowSplit	0.5


 Example 1

Example 2.

Parameter	Value
Honest Fees	100
Attacker Fees	0
GT Reward	50
Cost/GT	40


| PaySplit | 0.5 |

| PowSplit | 0.5 |

 Example 2

Example 3.

Parameter	Value
Honest Fees	100
GT	100
Attacker Fees	100
Cost/GT	26
Depth	2

 Example 3

Discussion

In Example 1 we can see that as the miner increases their hashing toward an average of 1 golden ticket per block, their losses are reduced. Importantly the reward for the attack is *always* negative.

In Example 2 the attacker is simply mining, not attacking. In this instance the optimum reward is simply the difference between the reward for the golden ticket and the cost of producing one. Given difficulty will push toward mining a golden ticket every second block, an equilibrium will be reached where mining is profitable.

In Example 3, we can see that an attack would become profitable if the golden ticket cost falls below 26. Honest mining would also be profitable here. Defending the network would simply require mining profitably such that difficulty pushes the cost per golden ticket above this threshold.

In all instances the network remains secure, with attacks on the mechanism simply defended. At the same time the mechanism can find a profitable equilibrium for honest operators.

Further Works

Finiding a generalised proof of the security of the network would be prove interesting and fruitful work.