A note on optimal double spending attacks

Juri Hinz and Peter Taylor

Abstract In the present note we address the important problem of stability of blockchain systems. The so-called ”double-sepending attacks” (attempts to spend digital funds more than once) have been analyzed by several authors. We re-state these questions under more realistic assumptions than previously discussed and show that they can be formulated as an optimal stopping problem.

Key words: blockchain, distributed ledger, double spending attack.

1 Introduction

In recent years, novel concepts originating from the blockchain idea have gained popularity. Their rapidly emerging software realizations are based on a mixture of traditional techniques (peer-to-peer networking, data encryption) and more modern concepts (consensus protocols). Digital currencies represent assets of these systems, their transactions are written and kept in an electronic ledger as part of the operation of the blockchain system. The main difference from a traditional financial system is that crypto-currencies are not issued and supervised by a central authority, but are maintained by joint efforts of a network consisting of independent computers (all running the same/similar software). Such a network searches for consensus which yields a common version of the ledger shared by all participants. The consensus is reached by means of a process, which is called *mining* and is usually backed by economic incentives. Blockchain systems are believed to achieve the same level of certainty and security as those governed by a central authority but do that at

Juri Hinz, e-mail: Juri.Hinz@uts.edu.au
School of Mathematical and Physical Sciences, University of Technology Sydney, PO Box 123, Broadway, NSW 2007, Australia.
Peter Taylor, e-mail: taylorpg@unimelb.edu.au
School of Mathematics and Statistics, University of Melbourne, VIC 3010, Australia.

significantly lower costs. Furthermore, because of the distributed, decentralized, and homogeneous architecture of the network, a blockchain system can reach a very high level of stability due to data redundancy and hard/software replicability.

Following the mining process, all network participants append, validate, and mutually agree on a common version of the data history, which is usually referred to as the *blockchain ledger*. Although the invention of mining is considered to be a real break-through which solves the long-standing consensus problem in computer science, there is criticism of this approach. The problem is that to reach consensus, real physical resources/efforts must be spent or at least allocated. For instance, the traditional Bitcoin protocol requires participants to solve cryptographic puzzles with real consumption of computing power and energy. This process is referred to as the so-called *proof of work*. Other blockchain systems avoid resource consumption and require temporary allocation of diverse resources, for instance the ownership of the underlying digital assets (proof of stake) or their spending (proof of burn). Furthermore, commitment of storage capacity (proof of storage) or a diverse combination of resource allocation/consumption can also be used.

Let us briefly elaborate on the proof of work; more details can be found in the excellent book by "Mastering Bitcoin" [1] by Andreas Antonopoulos. We focus on the Bitcoin protocol which was been initiated by [4], with a refinement on the double-spending problem in [5] and later in [3] with further considerations addressing propagation delay in [2]. In this framework, the ledger consists of a chain of blocks and each block contains valid transactions. The nodes compete to add a new block to the chain, and while doing so, each node attempts to collect transactions and solve a cryptographic puzzle. Once this puzzle is solved, it is made public to other nodes. This protocol also prescribes that if a peer node reports a completed block, then it must be verified, and if this block is valid, it must be attached to the chain, all uncompleted blocks shall be abandoned and a new block continuing the chain must be started. However, even following these rules, the chain forks regularly, which results in different nodes working on different branches. In order to reach a consensus in such cases, the protocol prescribes that a branch with shorter length must be abandoned as soon as a longer branch becomes known.

2 The double-spending problem

Now we return to the resilience of the protocol to attacks. Note that within a blockchain system, the nodes are running publicly available open-source software (for mining) which can easily be modified by any private user to control the computer nodes in order to undermine the system. In principle, there are may ways of doing this. One of the most obvious among malicious strategies would be an attempt to spend the electronic money more than once. The analysis of such a strategy is referred to as the double-spending problem.

In the classical [4], [5] formulation of this problem a merchant waits for $n \in \mathbb{N}_0$ confirming blocks after a payment by a buyer, before providing a product or service.

While the network is mining these n blocks, the attacker tries to build his/her own secret branch containing a version of the history in which this payment is not made. The idea is to not include the paying transaction in the private secret branch whose length will overtake the official branch to be then published. If this strategy succeeds, then the private secret branch becomes official and the payment disappears in the ledger after the product/service is taken by the attacker. Nakomoto [4] provides and Rosenfeld [5] refines an estimate of the attacker's success probability depending on his/her computational power and the number n of confirming blocks.

Let us briefly discuss their result before we elaborate on further details. In the framework of double-spending problem, it is assumed that a continuous-time Markov chain taking values in $\mathbb{Z}$ describes the difference in blocks between the official and secret branches. As in [5], we consider this process at time points at which a new block in one of the branches is completed, which yields a discrete-time Markov chain $(Z_t)_{t=0}^{\infty}$. Having started secret mining after the block including the attacker's payment (at block time $t=0$, $Z_0=0$) the attacker considers the following situation: At each time $t=1,2,3\ldots$, a new block in one of the branches (official or secret) is found, the block difference changes by ± 1 with probabilities (see [5], [2])

$$\mathbb{P}(Z_t = z+1\,|Z_{t-1}=z) = 1-q$$
$$\mathbb{P}(Z_t = z-1\,|Z_{t-1}=z) = q.$$

where $q \in]0,1[$ is the ratio of the computational power controlled by the attacker to the total mining capacity. Consider a realistic case where the attacker controls a smaller part of the mining power $0<q<1/2$ than that controlled by honest miners. In this case, if at any block time $t=0,1,2,\ldots$ the block difference is $z\in\mathbb{Z}$, then the probability $a(z,\infty)$ that the secret branch overtakes the official branch within unlimited time after t is given by

$$a_\infty(z,q) = \mathbb{P}(\min_{u=0}^{\infty} Z_u < 0\,|\,Z_0 = z) = \begin{cases} 1 & \text{if } z<0 \\ (\frac{q}{1-q})^{z+1} & \text{otherwise.} \end{cases} \tag{2.1}$$

Furthermore, at the time when the n-th block in the official branch is mined, the probability that the attacker has mined $m=0,1,2,\ldots$ blocks follows the *negative binomial distribution* whose distribution function is given by

$$F_{q,n}(k) = \sum_{m=0}^{k} \binom{n+m-1}{m} (1-q)^n q^m, \qquad k=0,1,2,\ldots. \tag{2.2}$$

Both results (2.1) and (2.2) are combined in [5] to obtain the success probability of the double-spending as follows: Consider the situation where at the time the n-th block in the official chain is completed, the attacker has mined $m>n$ blocks which can be published immediately. The probability of this event is given by

$$\sum_{m=n+1}^{\infty} \binom{n+m-1}{m} (1-q)^n q^m = 1 - \sum_{m=0}^{n} \binom{n+m-1}{m} (1-q)^n q^m = 1 - F_{q,n}(n).$$

Next, consider the opposite event, assuming that when the n-th official block is completed, the attacker has not overtaken the official chain in which case $m \leq n$. In this case, the probability of winning the race is given by

$$\begin{aligned} \sum_{m=0}^{n} \binom{n+m-1}{m} (1-q)^n q^m a_\infty(n-m,q) &= \frac{q}{1-q} \sum_{m=0}^{n} \binom{n+m-1}{m} (1-q)^m q^n \\ &= \frac{q}{1-q} F_{1-q,n}(n). \end{aligned}$$

Clearly, the success probability is given by

$$1 - F_{q,n}(n) + \frac{q}{1-q} F_{1-q,n}(n). \tag{2.3}$$

For instance, if the merchant waits for six confirming blocks the attack succeeds with probabilities

$$0.00037\% \qquad \text{for } q = 6\%, \quad \text{and with} \quad 0.0025\% \quad \text{for } q = 8\%.$$

As a result, waiting for six blocks after the payment has been considered as secure in the sense that with realistic efforts it is practically impossible to succeed with double spending.

Remark: Note that in the original work [5] it was assumed that the attacker can start the race having pre-mined one block. This leads to a different success probability

$$1 - F_{q,n}(n-1) + F_{1-q,n}(n-1). \tag{2.4}$$

While the difference between (2.4) and (2.3) can be significant (see Figure 1), it is not clear how to achieve an advantage of being able to start the race with one block ahead of the official chain. The present note is devoted to this interesting question.

The above analysis [5] calculates the probability of the alternative blockchain getting ahead of the official one. It doesn't consider revenues and losses from a successful/failed attack. Furthermore, the possibility of canceling the secret mining (if the block difference becomes too high) is not considered. Most important, however, is the question why the paying transaction must be placed right after fork-off. Note that this assumption is justifiable only if the merchant requires immediate payment after the purchase is agreed upon, otherwise canceling the deal. However, in reality, the attacker may be able to freely choose the time of payment, in particular when buying goods from web portals. That is, an attempt to overtake the official chain before launching an attack can give an advantage in the spirit of the above remark.

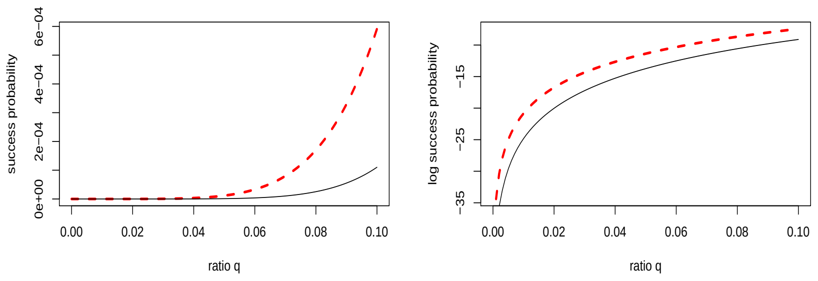

Fig. 1 The success probability (and its logarithm) of the double spending attack for $n = 6$ confirming blocks depending on the mining ratio $q \in [0, \frac{1}{10}]$ calculated by (2.3) (solid line) versus (2.4) (dashed line).

3 A refinement of the double-spending problem

Let us consider an alternative situation. Assume that the attacker can freely choose the time of payment. Doing so, he/she can start working on a private secret branch long before the payment is placed. For such situations, the analysis of the double-spending is different and requires solving (multiple) stopping problems.

Consider a finite time horizon where $t \in \{0,\dots,T\}$ represents the number of blocks mined in the official chain since the branch has forked off. That is, we suppose that our secret mining starts at the block time $t = 0$. We interpret $T \in \mathbb{N}_0$ as the maximal length of the official branch, which can be abandoned if a longer branch has been discovered. To the best of our knowledge, the current Bitcoin protocol does not have such a restriction, meaning that the shorter branch must always be discarded, independently of its length . However, other blockchain systems discuss ”checkpoints” and ”gates” with a similar functionality. A finite time horizon yields conceptual advantages and presents a negligible deviation from the reality since T can be sufficiently large.

Recall the process $(Z_t)_{t=0}^{\infty}$, describing the branch length difference at times the next block in one of the branches has been mined. Now, consider another process $(X_t)_{t=0}^{\infty}$, where X_t stands for the branch length difference between the official and secret branches at the times $t = 0,1\dots$, when one new block in the *official* branch is completed. In turns out that $(X_t)_{t=0}^{\infty}$ follows a Markov chain, whose transition from state $X_t = x$ to $X_{t+1} = x+1-y$ describes the event that while one official block has been mined, $y = 0,1,2,\dots$ secret blocks have been obtained. The transition probabilities of $(X_t)_{t=0}^{T}$ are given for all $t = 0,1\dots$, by

$$\mathbb{P}(X_{t+1} = x+1-k | X_t = x) = \begin{cases} G(k), & k \in \mathbb{N}_0, \\ 0, & k \in \mathbb{Z} \setminus \mathbb{N}_0, \end{cases} \tag{3.1}$$

in terms of the geometric distribution

$$G(k) = (1-q)q^k \quad \text{for } k \in \mathbb{N}_0 = \{0,1,2,\dots,\}. \tag{3.2}$$

Suppose that the secret branch contains the invalidation of the paying transaction. Recall this can be reached by a simple non-inclusion of the attacker's paying transaction. If the attack is launched at a block time $\tau = \{0,\dots,T\}$, then the payment will be included into block $\tau+1$ of the official branch. In this context, the crucial question is whether to attack or not and how to chose the time $\tau = 0,\dots,T$ optimally. It turns out that under specific assumptions, this question can be treated as an optimal stopping problem, which we formulate next.

According to our modeling with a finite time horizon, we agree that for $\tau > T-n$ a successful attack is not possible. Namely, since the payment is placed into block $\tau+1$ and n confirming blocks are expected, the last confirmation block $\tau+n > T$ would be beyond the maximal branch length which can be abandoned. That is, we can assume that the time τ must be chosen within the finite horizon $\tau = 0,\dots,\tilde{T}$ with the last time point $\tilde{T} = T-n$. The decision whether to attack must be based on the current block time $t = 0,\dots,\tilde{T}$ and on the recent block difference X_t. In order to optimize the time $\tau = 0,1,\dots,\tilde{T}$, we define the success event $S(\tau)$ for the attack launched at τ as

$$S(\tau) = \{\min_{i=\tau+n+1}^{T+1} X_i \le 0\} \quad \tau = 0,\dots,\tilde{T}.$$

Hence the expected reward of the attack is

$$\begin{aligned} R_\tau(x) &= \mathbb{E}(C1_{S(\tau)} - c1_{S(\tau)^c} \,|\, X_\tau = x) \\ &= (C+c)\mathbb{P}(S(\tau)|X_\tau = x) - c, \quad \tau = 0,\dots,\tilde{T},\ x \in \mathbb{Z} \end{aligned} \tag{3.3}$$

where the numbers $C > 0$ and $c > 0$ represent the revenue and loss resulting from the success or failure of the attack. Let us agree that $\tau = +\infty$ stands for the attacker's option to not attack, which can be optimal if the chance of overtaking the official branch is too low. In order to model such an opportunity, we extend the reward function (3.3) for the time argument $t = \infty$ as

$$R_\infty(x) = 0, \quad x \in \mathbb{Z}. \tag{3.4}$$

In this context, the choice of the optimal payment time τ^* yields an optimal stopping problem of the following type:

$$\begin{gathered} \text{determine a maximizer } \tau^* \text{ to } \mathscr{T} \to \mathbb{R}, \quad \tau \mapsto \mathbb{E}(R_\tau(X_\tau)) \text{ where} \\ \mathscr{T} \text{ denotes all } \{0,\dots,\tilde{T}\} \cup \{+\infty\}\text{-valued stopping times.} \end{gathered} \tag{3.5}$$

4 Conclusion

Having assumed that the payment moment can be chosen by the attacker, the solution to the double spending problem consists of secret mining, followed by a later payment. The optimal payment time is determined by the current numbers of blocks in both chains since their fork off. The success probability of such an attack is dependent on the required confirmation block number n and the revenue/loss $C > 0$, $c > 0$ caused by the success/failure of the attack. This contribution shows that the optimization of the attack under these assumptions requires solving an optimal stopping problem. The authors will address these problems in future research.

Acknowledgements The first author expresses his warmest gratitudes to Kostya Borovkov for useful remarks.

References

1. Andreas M. Antonopoulos. *Mastering Bitcoin: Programming the Open Blockchain*. O'Reilly Media, Inc., 2nd edition, 2017. ISBN 1491954388, 9781491954386.
2. J. Goebel, H.P. Keeler, A.E. Krzesinski, and P.G. Taylor. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Performance Evaluation*, 104 (Supplement C): 23 – 41, 2016. ISSN 0166-5316. URL http://www.sciencedirect.com/science/article/pii/S016653161630089X.
3. C. Gruenspan and R. Perez-Marco. Double spend races. *Working paper*, 2017.
4. S. Nakomoto. A peer-to-peer electronic cash system. *Working paper*, 2008.
5. M. Rosenfeld. Analysis of hashrate-based double spending. *Working paper*, 2014.