Upper Bound Probability of Double Spend Attack on SPECTRE

Lyudmila Kovalchuk
Input Output HK,
National Technical University of
Ukraine “Igor Sikorsky Kyiv
Polytechnic Institute”
Kyiv, Ukraine
lyudmila.kovalchuk@iohk.io

Mariia Rodinko
Input Output HK,
V. N. Karazin Kharkiv
National University
Kharkiv, Ukraine
mariia.rodinko@iohk.io

Roman Oliynykov
Input Output HK,
V. N. Karazin Kharkiv
National University
Kharkiv, Ukraine
roman.oliynykov@iohk.io

ABSTRACT
We obtained analytical expressions of upper estimates for success probability of a double spend attack on DAG-based consensus protocol SPECTRE, depending on network parameters. Using such estimates, it is possible to evaluate the number of confirmation blocks that is sufficient for prevention of such attacks.

CCS CONCEPTS
• Security and privacy → Distributed systems security.

KEYWORDS
blockchain, directed acyclic graph, proof-of-work, SPECTRE, double spend attack

ACM Reference Format:

1 INTRODUCTION
The new generation of consensus protocols are based on the directed acyclic graph (DAG) structure. Such protocols are SPECTRE [5], PHANTOM [6], Graphchain [1], Tangle [4] and other proposals, where blocks or transactions form a DAG as a distributed ledger. The main objective is to maximize the utilization of the network capacity having strong security guarantees. Some characteristics of the Nakamoto consensus were really improved by these protocols, resulting in higher throughput and faster transaction confirmation, but not providing properties, like linear block ordering, in full.

In this paper, we will consider SPECTRE protocol [5]. It is one of the earliest and fastest DAG protocols with rather unusual consensus algorithm. In SPECTRE, a newly created block is published immediately, and it refers to all recent blocks in DAG. A protocol provides a pairwise voting algorithm that is carried out by all blocks. The value of a vote is defined by the block position in DAG.

SPECTRE provides fast confirmation of blocks and good scalability, but only weak liveness. The authors state that this protocol is secure against different attacks, such as double spend and censorship, but give no proofs for this statement, only some empirical considerations.

In what follows, we propose some hybrid of censorship and double spend attacks. During this attack malicious miners not only generate alternative chains or subDAGs, but also choose which blocks to confirm. Such an attack is more effective than both mentioned attacks separately. We also show that this attack can be resisted (under certain restrictions on the fraction of an adversary’s computational power) if the rules of transactions’ acceptance by vendor are changed a little. We also present rigorously substantiated numerical estimates of probability of such attack depending on the network parameters and the number of the confirmation blocks.

2 SPECTRE DESCRIPTION
SPECTRE [5] is a Proof-of-Work protocol where blocks are ordered in a direct acyclic graph (DAG) \( G = (C, E) \), where \( C \) are blocks, \( E \) are hash references. The rules for miners are simple [1]:

1. When creating or receiving a block, transmit the block to all peers.
2. When creating a block, embed in its header a list containing the hash of all leaf-blocks (blocks with in-degree 0) in the locally-observed DAG.

To avoid conflicting transactions, a voting procedure is used. Every block \( z \in G \) is considered a voter. To describe the voting rules, some definitions should be introduced [5]:

- \( \text{past}(z, G) \subseteq C \) denotes the subset of blocks reachable from \( z \);
- \( \text{future}(z, G) \subseteq C \) denotes the subset of blocks from which \( z \) is reachable;
- \( \text{cone}(z, G) \) denotes the set of blocks that the DAG directly orders with respect to \( z \);
- \( \text{virtual}(G) \) denotes a hypothetical block that satisfies \( \text{past}(\text{virtual}(G)) = G \).

If there are two blocks \( x, y \) that have conflicting transactions, voting should be used to decide \( x \prec y \) (vote -1) or \( y \prec x \) (vote +1) (if there is a tie, then a vote is 0). The voting rules are as follows [5]:

1. If \( z \in G \) is in \( \text{future}(x) \) but not in \( \text{future}(y) \) then it will vote in favour of \( x \) (i.e., for \( x \prec y \)).
2. If \( z \in G \) is in \( \text{future}(x) \cap \text{future}(y) \) then \( z \)'s vote will be determined recursively according to the DAG that is reduced.
to its past, i.e., it has the same vote as virtual(past(z)). If the result of this vote is a tie, z breaks it arbitrarily.
(3) if z ∈ G is not in the future of either blocks then it will vote the same way as the vote of the majority of blocks in its own future.
(4) if z is the virtual block of G then it will vote the same way as the vote of the majority of blocks in G.
(5) finally, (for the case where z equals x or y), z votes for itself to succeed any block in past(z) and to precede any block outside past(z).

There are two types of attacks analyzed in the original paper: doublespending and censorship [5]. In a doublespending attack, the first mining rule is broken (an attacker builds blocks in secret and publishes two chains with conflicting transactions together). In a censorship attack, an attacker breaks the second rule and does not build on all fresh blocks, he mines only on those blocks that are included into the tree which contains an alternative transaction).

We present below a hybrid attack, where two rules are broken by an attacker at the same time.

3 DESCRIPTION OF THE ATTACK

All stages of this attack are given on Figure 1 where:

- T₁ is the moment, when malicious miners generated CM₁ and start to generate CM₂;
- T₂ is the moment, when malicious miners publish CM₂;
- T₃ is the moment, when malicious miners receive goods and publish CM₂ ∪ CM₃.

![Figure 1: An example of the hybrid attack](image)

To simplify Figure 1, we picture a definite number of blocks on it. But in the proof, we will refer to an arbitrary number of blocks in each chain. Also we need to note that when we replace any chain by a “tree” with the same number of blocks, nothing changes in the model of attack and in the proof of the theorem.

Stage I.

- Honest miners generate the chain CH₁ of the length h₁.
- Malicious miners generate a block with transaction X₀ and confirm this block, i.e. generate the chain CM₁ of the length m₁ (but do not open this chain).

Stage II.

Starts from the moment T₁, when malicious miners decide that they generate enough confirmation blocks for a transaction X₀.
- Malicious miners generate a block with a transaction Y₀ and confirm this block, i.e. generate the chain CM₂ of the length m₂ (but do not open this chain).
- Honest miners generate the chain CH₂ of the length h₂, which continues the chain CH₁.

Stage III.

Starts from the moment T₂, when malicious miners decide that they generate enough confirmation blocks for transaction Y₀.
- Malicious miners open the chain CM₁.
- Vendor waits for some time, and meanwhile honest miners generate the chain CH₃ of the length h₃ ≥ z for some appropriate z, and then sends goods or services.
- Meanwhile malicious miners generate the chain CM₃ of the length m₃ that continues to confirm the block with X₀. Malicious miners still do not open the chains CM₂ and CM₃.

Stage IV.

Starts from the moment T₃, when the vendor sends goods or services.
- Vendor sends the goods at the moment T₃.
- Immediately after that, malicious miners open two chains CM₂ and CM₃.
- Honest miners confirm both chains, CH₃ and CM₃, with the chain CH₄ of the length h₄.
- Malicious miners confirm only the chain CM₃ with the chain CH₄ of the length m₄ (combining double spend and censorship attacks).

Under these conditions and according to the voting protocol of SPECTRE, the voting procedure will be as follows:

- all the blocks of CM₁ vote for X₀ (m₁ blocks);
- all the blocks of CH₁ vote for X₀ (h₃ ≥ z blocks);
- all the blocks of CM₂, CM₃, CM₄ vote for Y₀ (m₂ + m₃ + m₄ blocks);
- all the blocks of CH₁ and CH₂ vote as the majority in their futures (h₁ + h₂ blocks);
- all the blocks of CH₄ vote as the majority in their pasts (h₄ blocks).

It should be noted that the vendor’s behavior in this case should be different from the standard one, namely: he should wait for no less than z blocks after the moment he sees the transaction X₀ for the first time, even if at the time of its first appearance the block containing this transaction has no less than z descendants. This requirement will not allow an adversary to attack with the probability close to 1.

Now we are going to prove the theorem on the upper bound of the success probability of a double spend attack. In what follows, we need some designations and notations from [2, 3].

Let α₄, α₃ > 0 be some values that characterize the intensity of block generation for honest miners and malicious miners, respectively, and α = α₄ + α₃ is the general intensity. We define these values as the parameters of distribution functions:

\[
F_{TH} = P(T_H < t) = 1 - e^{-\alpha_H t},
F_{TM} = P(T_M < t) = 1 - e^{-\alpha_M t},
\]

where T_H, T_M are random variables that measure the time it takes to mine a block for honest miners and malicious miners, respectively.
Then the probability that honest miners generate the next block before malicious miners is 
\[ p_H = \frac{\alpha_H}{\alpha_M + \alpha_H}, \quad p_M = 1 - p_H = \frac{\alpha_M}{\alpha_M + \alpha_H} \]  

(2)

We also assume that \( D_H \) denotes the time it takes for honest miners to share a block (after it was generated) for all (at least all honest) nodes in network; \( D_M \) is the respective time for malicious miners, and here we assume \( D_M = 0 \), i.e. an adversary is well-synchronized.

Also let us define: 
\[ p'_H = \text{the probability of the event that the honest miners generate and share the next block for all (at least honest) nodes before malicious miners do this}, \]
\[ p'_M = 1 - p'_H \text{ is the probability of the alternative event.} \]

Then, according to Lemma 1, 
\[ p'_H = e^{-\alpha_M D_H} p_H \]
\[ p'_M = 1 - e^{-\alpha_M D_H} p_H. \]

(3)

Designate as \( P_z(k) \) the probability that malicious miners generate exactly \( k \) blocks until honest miners generate and share \( z \) blocks. Then, according to Lemma 4 in [3], 
\[ P_z(k) = \sum_{i=0}^{k} \binom{z+i-1}{i} p_H^i p_M^{k-i} e^{-\alpha_M n D_H} \]
\[ \times \frac{(\alpha_M n D_H p_M)^{k-i}}{(k-i)!}. \]

(4)

Note that in the sum (4) only one summand is left: for \( k = i \); all others vanish because of \((\alpha_M n D_H p_M)^{k-i}\).

Now we are ready to formulate the main result.

**Theorem 3.1.** Let the vendor wait for \( z \) blocks after he saw for the first time the transaction \( X_0 \). Then, in the notations (1)-(5), the upper bound of the probability \( P_z(\alpha_M, \alpha_H, D_H) \) of the success by the MM is: 
\[ P_z(\alpha_M, \alpha_H, D_H) \leq \]
\[ \leq 1 - \sum_{k=0}^{z-1} P_z(k) \cdot \left( 1 - \frac{p'_M}{p_H} \right)^{z-k} \left( \left( p'_H \right)^{z-k-1} + 1 \right) \]
\[ \leq 1 - \sum_{k=0}^{z-1} P_z(k) \cdot \left( 1 - \frac{p'_M}{p_H} \right)^{z-k} \left( \left( p'_H \right)^{z-k-1} + 1 \right). \]

(6)

and in the particular case when \( D_H = 0 \) 
\[ P_z(\alpha_M, \alpha_H, 0) \leq \]
\[ \leq 1 - \sum_{k=0}^{z-1} P_z(k) \cdot \left( 1 - \frac{p'_M}{p_H} \right)^{z-k} \left( \left( p'_H \right)^{z-k-1} + 1 \right). \]

(7)

Let us analyze votes in the past \( (B_i) \):
- all blocks from \( CH_2 \) vote for \( X_0 \);
- all blocks from \( CM_3 \) vote for \( Y_0 \);
- the last block of \( CH_2, B_2 \) (in Figure 1 after the block 11), votes for \( Y_0 \) if \( h_3 \leq m_3 \) and so will all blocks from \( CH_1 \) and \( CH_2 \).

Therefore one of conditions of a successful attack is \( h_3 \leq m_3 \) that we designate as the condition C1:

\[ h_3 \leq m_3. \]

When (C1) does not hold, malicious miners still may have success in the case if blocks in the past \( (B_2) \) vote for \( Y_0 \). Let us analyze when it is possible, if C1 does not hold, then \( B_2 \) votes for \( X_0 \). Let \( h_3 = m_3 + l \). Then the block that is just before \( B_2 \), named \( B_3 \) (in Figure 1 this is 10), will vote for \( Y_0 \) only if during the time between \( B_3 \) and \( B_2 \), malicious miners generate not less than \( l + 1 \) blocks.

Similarly, if \( B_3 \) and \( B_2 \) vote for \( X_0 \), the necessary condition for the previous block \( B_4 \) to vote for \( Y_0 \) is that during the time between \( B_4 \) and \( B_3 \) malicious miners generate not less than \( l + 2 \) blocks, and so on.

So, if C1 does not hold, the necessary condition for the event "some block number \( i \) in the chain \( CH_2 \) votes for \( Y_0 \)" is the condition C2:

\[ \exists B_5 \in CH_2 \text{ that during the period between } B_5 \text{ and the next block malicious miners generate not less than } l + 1 + u \text{ blocks, where } u \text{ is the number of blocks in } CH_2 \text{ that are in } future(B_5). \]

If C1 and C2 do not hold, malicious miners still have the chance to realize the attack. The necessary condition for this is C3:

"at some moment \( T > T_3 \) the following inequality holds: \( m_4 \geq h_4 + \ell \)."

So, for success of an attack at least one of the conditions C1, C2, C3 has to hold.

Note that we assume that block generation corresponds to Poisson process, so block generations on different time intervals and on different branches (or subDAGs) are independent. Then the events C1, C2 and C3 are independent.

According to the condition of theorem, \( h_3 \geq z \). Then, using (4) and Lemma 4 in [3] we get that 
\[ P(C1) = 1 - \sum_{k=0}^{z-1} P_z(k). \]

(8)

Next, 
\[ P(C1 \text{ does not hold}, C2 \text{ holds}) = \]
\[ = P(C1 \text{ does not hold}) \times P(C2 \text{ holds}) = \]
\[ = \sum_{k=0}^{z-1} P_z(k) \left( p'_M \right)^{z-k} + \left( p'_M \right)^{z-k+1} + ... + \left( p'_M \right)^{z-k+(k_2-1)} \]
\[ \leq \sum_{k=0}^{z-1} P_z(k) \frac{\left( p'_M \right)^{z-k}}{1 - p'_M} = \sum_{k=0}^{z-1} P_z(k) \frac{\left( p'_M \right)^{z-k}}{p_H}. \]

(9)

The probability 
\[ P(C1 \text{ and } C2 \text{ does not hold}, C3 \text{ holds}) = \]
\[ = (1 - P(C1))(1 - P(C2)) \cdot P(C3) \leq \]
\[ (1 - P(C1)) \cdot P(C3). \]
Note that the condition C3 means “malicious miners will catch up from at least \( z - k \) blocks behind”, so
\[
P(C3) \leq \left( \frac{\hat{P}'_M}{\hat{P}_H} \right)^{z-k}.
\]
Then
\[
P(C1 \text{ and } C2 \text{ does not hold, } C3 \text{ holds}) \leq \sum_{k=0}^{z-1} P_z(k) \left( \frac{\hat{P}'_M}{\hat{P}_H} \right)^{z-k}.
\]
So, from (10) - (8), we obtain
\[
P_z(\alpha_M, \alpha_H, D_H) \leq 1 - \sum_{k=0}^{z-1} P_z(k) \left( \frac{\hat{P}'_M}{\hat{P}_H} \right)^{z-k} + \sum_{k=0}^{z-1} P_z(k) \left( \frac{\hat{P}'_M}{\hat{P}_H} \right)^{z-k} = 1 - \sum_{k=0}^{z-1} P_z(k) \left[ 1 - \left( \frac{\hat{P}'_M}{\hat{P}_H} \right)^{z-k} \right] = 1 - \sum_{k=0}^{z-1} P_z(k) \left[ 1 - \left( \frac{\hat{P}'_M}{\hat{P}_H} \right)^{z-k} \left( \left( \frac{\hat{P}'_M}{\hat{P}_H} \right)^{z-k} - 1 \right) \right].
\]
The theorem is proved.

Tables 1 and 2 give the minimal number of confirmation blocks for which the probability of attack is not greater than \( 10^{-3} \), for various network parameters (calculated using (6)). We calculated these probabilities for two different values of general intensity \( \alpha \): for \( \alpha = \frac{1}{600} = 0.00167 \), as for BTC, and for \( \alpha = \frac{1}{60} = 0.0167 \), or 10 times bigger than for BTC.

Also note that these Tables are essentially different form the Tables 4 and 5 in [3], which also give the number of confirmation blocks for different network parameters. The matter is that in [3] we considered “classical” blockchain while in this paper we consider DAG with more complicated consensus protocol.

### Table 1: The minimal number \( z \) of confirmation blocks for which \( P_z(\alpha_M, \alpha_H, D_H) \leq 10^{-3} \) for various network parameters \( \Delta \) (in seconds) and \( p_M \), and for \( \alpha = 0.00167 \) (as for BTC)

<table>
<thead>
<tr>
<th>PM</th>
<th>0 sec</th>
<th>15 sec</th>
<th>30 sec</th>
<th>60 sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>0.15</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>0.2</td>
<td>14</td>
<td>14</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>0.25</td>
<td>21</td>
<td>21</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>0.3</td>
<td>33</td>
<td>35</td>
<td>36</td>
<td>40</td>
</tr>
<tr>
<td>0.35</td>
<td>60</td>
<td>65</td>
<td>69</td>
<td>81</td>
</tr>
<tr>
<td>0.4</td>
<td>137</td>
<td>154</td>
<td>174</td>
<td>228</td>
</tr>
</tbody>
</table>

### Table 2: The minimal number \( z \) of confirmation blocks for which \( P_z(\alpha_M, \alpha_H, D_H) \leq 10^{-3} \) for various network parameters \( \Delta \) (in seconds) and \( p_M \), and for \( \alpha = 0.0167 \)

<table>
<thead>
<tr>
<th>PM</th>
<th>0 sec</th>
<th>15 sec</th>
<th>30 sec</th>
<th>60 sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>6</td>
<td>7</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>0.15</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td>0.2</td>
<td>14</td>
<td>17</td>
<td>23</td>
<td>43</td>
</tr>
<tr>
<td>0.25</td>
<td>21</td>
<td>29</td>
<td>44</td>
<td>172</td>
</tr>
<tr>
<td>0.3</td>
<td>33</td>
<td>55</td>
<td>114</td>
<td>1</td>
</tr>
<tr>
<td>0.35</td>
<td>60</td>
<td>159</td>
<td>172</td>
<td>1</td>
</tr>
<tr>
<td>0.4</td>
<td>137</td>
<td>203</td>
<td>228</td>
<td>1</td>
</tr>
</tbody>
</table>

To calculate the number of confirmation blocks for Tables 1 and 2, we used formulas (6), (7) and calculated
\[
\min_{z \geq 1} P_z(\alpha_M, \alpha_H, D_H) \leq 10^{-3},
\]
for fixed \( \alpha_M, \alpha_H \) and \( D_H \).

We chose \( D_H \) as 0 \( \leq D_H \leq 60 \) sec that is wider than range of values of \( D_H \) which occurs in practice: for example, for BTC, the most often values for \( D_H \) are from 0 to 20 sec.

We should note that the first rows (for \( D_H = 0 \)) in Table 1 and Table 2 are identical, while all other rows are very different. The explanation of this fact is the next: the larger time delay \( D_H \) for honest miners, the larger advantage exists for malicious miners. Moreover, when \( D_H \neq 0 \), this advantage essentially depends on the general intensity \( \alpha \) of block generation: for large \( \alpha \) malicious miners have a large advantage. This fact was proved analytically in [3], and we can see it in formulas (3): the larger \( \alpha_M \cdot D_H \), the larger \( \hat{P}'_M \); the larger \( \hat{P}'_M \), the smaller security threshold (for \( D_H = 0 \) the security threshold is 50%). For example, as we can see from the last row of the Table 2, for \( D_H = 60 \) sec and \( \alpha = 0.0167 \), the security threshold is not more than 0.3, which means that under these parameters malicious miners can succeed even with ratio 30%.

### 4 CONCLUSIONS
Several consensus protocols based on DAG were proposed in last 5-7 years. It should be noted that implementation of these protocols, though, increases efficiency of transactions processing, but carries some risks. There were no rigorous proof (from the mathematical point of view) of resilience to splitting and double spend attacks for any of DAG-based protocols. Constructing of such proofs is an analytically difficult task, even in the case of using of a very simplified mathematical model of network operation.

In this paper, we make a first step in construction of security estimates of DAG-based consensus protocols, building the upper bounds of a double spend attack probability for SPECTRE protocol. To increase the security of SPECTRE consensus protocol, we change a bit a standard vendor’s behavior: he should wait for \( z \) confirmation blocks which must appear under his observation. The important property of the obtained estimates is the possibility to use them for calculation of both specific values of attack success probability and of the required number of confirmation blocks, depending on the network parameters.
The method for building such estimates essentially depends on consensus protocol itself: it is suitable for SPECTRE but not for all DAG protocols. But some ideas from the method might help to build such estimates for other DAGs.

REFERENCES