Superlight Blockchain Client with Minority Upgrade

Anonymous Author(s)

ABSTRACT

Superlight clients allow decentralized wallets to learn facts about the blockchain without downloading all the block headers. They achieve exponentially faster communication compared to SPV clients. For proof-of-work, they implement the so-called NIPoPoW primitive, for which there exist two variants: Superblock clients and FlyClient. Both of these protocols require consensus changes to existing blockchains and at least a soft fork to implement. In this paper, we discuss how a blockchain can be upgraded to support superblock clients without a soft fork. We show that it is possible to implement the needed changes without modifying the consensus protocol and by requiring only a minority of miners to upgrade, an upgrade termed a "velvet fork" in the literature. While previous work conjectured that NIPoPoW can be safely deployed using velvet forks as-is, we show that previous constructions are insecure, and that using velvet techniques to interlink a blockchain can pose insidious security risks. We describe a novel attack which we term a "chain-sewing" attack which is only possible in velvet situations: An adversary can cut-and-paste portions of various chains from independent forks, sewing them together to form a proof that looks like a chain but is not. We demonstrate that a minority adversary can thwart previous constructions with overwhelming probability. We put forth the first provably secure velvet NIPoPoW construction. Our construction is secure against adversaries that are bounded by 1/2 of the upgraded honest miner population. We prove our construction achieves persistence and liveness and analyze the trade-offs between the upgraded population parameter and the succinctness of the construction. Like non-velvet NIPoPoWs, our approach allows proving generic predicates about chains using infix proofs and as such can be adopted in practice for fast synchronization of transactions and accounts.

CCS CONCEPTS

• **Security and privacy** → Use https://dl.acm.org/ccs.cfm to generate actual concepts section for your paper;

KEYWORDS

blockchain, consensus, lightclient, NIPoPoW

1 INTRODUCTION

[4][5][1][2][3]

2 SUPERBLOCKS & VELVET FORK

2.1 Velvet fork parameter

A velvet fork suggest that only a minority of upgraded parties needs to support the protocol changes. Let g express the percentage of honest upgraded parties to the total number of miners. We will refer to g as the "velvet parameter".

Definition 1. Velvet Parameter *Let g be the velvet parameter for NIPoPoW protocols. Then if n_h the upgraded honest miners and n the total number of miners t out of which are corrupted, it holds that $n_h = g(n - t)$.*

2.2 Smooth and Thorny blocks

In order to be applied under velvet fork, a protocol has to change in a backwards-compatible manner. In essence, any additional information coming with the protocol upgrade is transparent to the non-upgraded players. This transparency towards the non-upgraded parties requires any block that conforms only to the old protocol rules to be considered a valid one. Considering superblock NIPoPoWs under a velvet fork, any block is to be checked for its validity regardless the validity of the NIPoPoWs protocol's additional information, which is the interlink structure.

A block generated by the adversary could thus contain arbitrary data in the interlink and yet be appended in the chain adopted by an honest party. In case that trash data are stored in the structure this could be of no harm for the protocol routines, since such blocks will be treated as non-upgraded. In the context of the attack that will be presented in the following section, we examine the case where the adversary includes false interlink pointers.

An interlink pointer is the hash of a block. A correct interlink pointer of a block b for a specific level μ is a pointer to the most recent b's ancestor of level μ. From now on we will refer to correct interlink pointers as *smooth pointers.* Pointers of the 0-level *(prevIds)* are always smooth because of the performed proof-of-work.

Definition 2. Smooth Pointer *Smooth pointer of a block b for a specific level μ is the interlink pointer to the most recent μ-level ancestor of b.*

A non-smooth pointer may not point to the most recent ancestor of level μ or even point to a superblock of a fork chain, as shown in Figure 1.

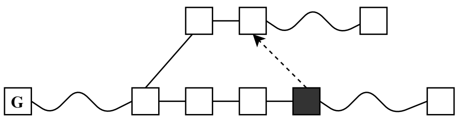

Figure 1: ***A non-smooth pointer of an adversarial block, colored black, in an honest player's chain.***

In the same manner it is possible that a false interlink contain arbitrary pointers to blocks of any chain as illustrated in Figure 2. The interlink pointing to arbitrary directions resembles a thorny bush, so we will refer to blocks containing false interlink information as *thorny.*

Definition 3. Thorny Block *Thorny block is a block which contains at least one non-smooth interlink pointer.*

Opposite of the thorny are the *smooth* blocks, which may be blocks generated by non-upgraded players or blocks generated

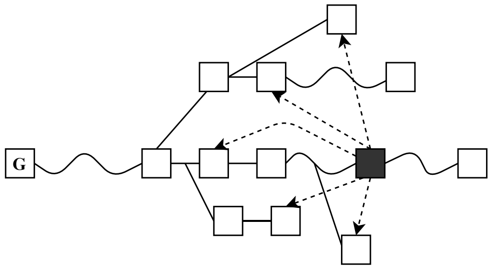

Figure 2: ***A thorny block appended in an honest player's chain. The dashed arrows are interlink pointers.***

by upgraded players and contain only smooth pointers in their interlink.

Definition 4. **Smooth Block** *Smooth block is any block which is not thorny.*

3 THE CHAINSEWING ATTACK

We will now describe an explicit attack against the NIPoPoW suffix proof construction under velvet fork. As already argued, since the protocol is implemented under velvet fork, any thorny block be accepted as valid. Taking advantage of such blocks in the chain, the adversary could produce suffix proofs containing an arbitrary number of blocks belonging in several fork chains. The attack is described in detail in the following.

Assume that chain C_B was adopted by an honest player B and chain $C_{\mathcal{A}}$, a fork of C_B at some point, maintained by adversary $\mathcal{A}$. Assume that the adversary wants to produce a suffix proof in order to attack a light client to have him adopt a chain which contains blocks of $C_{\mathcal{A}}$. In order to achieve this, the adversary needs to include a greater amount of total proof-of-work in her suffix proof, $\pi_{\mathcal{A}}$, in comparison to that included in the honest player's proof, π_B, so as to achieve $\pi_{\mathcal{A}} \geq_m \pi_B$. For this she produces some thorny blocks in chains $C_{\mathcal{A}}$ and C_B which will allow her to claim blocks of chain C_B as if they were of chain $C_{\mathcal{A}}$ in her suffix proof.

The general form of this attack for an adversary sewing blocks to one forked chain is illustrated in Figure 3. Dashed arrows represent interlink pointers of some level $\mu_{\mathcal{A}}$. Starting from a thorny block in the adversary's forked chain and following the interlink pointers, a chain is formed which consists the adversary's suffix proof. Blocks of both chains are included in this proof and a verifier could not distinguish the non-smooth pointers participating in this proof chain and, as a result, would consider it a valid proof.

As the generic attack scheme may seem a bit complicated we will now describe a more specific attack case. Consider that the adversary acts as described below. Assume that the adversary chooses to attack at some level $\mu_{\mathcal{A}}$. As shown in Figure 4 she first generates a superblock b' in her forked chain $C_{\mathcal{A}}$ and a thorny block a' in the honest chain C_B which points to b'. As argued earlier, block a' will be accepted as valid in the honest chain C_B despite the invalid interlink pointers. After that, the adversary may mine on chain $C_{\mathcal{A}}$ or C_B, or not mine at all. At some point she produces a thorny block a in $C_{\mathcal{A}}$ pointing to a block b of C_B. Because of the way blocks are generated by updated honest miners there will be successive interlink pointers leading from block b to block a'. Thus following the interlink pointers a chain is formulated which connects $C_{\mathcal{A}}$ blocks a and b' and contains an arbitrarily large part of the honest player's chain C_B.

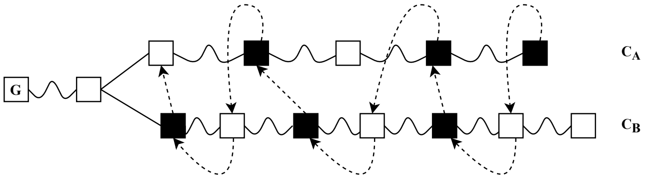

Figure 3: ***Generic Chainsewing Attack.*** C_B ***is the chain of an honest player and*** $C_{\mathcal{A}}$ ***the adversary's chain. Adversarially generated blocks are colored black. Dashed arrows represent interlink pointers included in the adversary's suffix proof. Wavy lines imply one or more blocks.***

At this point the adversary will produce a suffix proof for chain $C_{\mathcal{A}}$ containing the subchain $C\{ab\} \cup C\{b : a'\} \cup C\{a' : b'\}$. Notice that following the interlink pointers constructed in such a way, a light client perceives $C\{ab\} \cup C\{b : a'\} \cup C\{a' : b'\}$ as a valid chain.

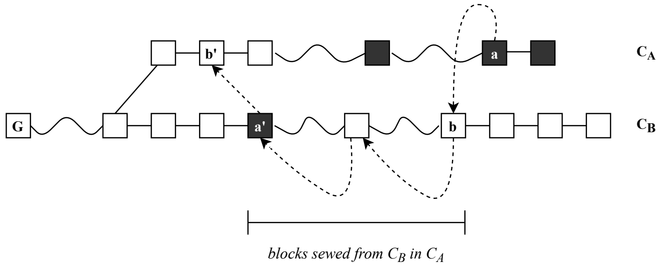

Figure 4: ***Chainsewing Attack.*** C_B ***represents the chain of an honest player.*** $C_{\mathcal{A}}$ ***is an adversarial fork. Adversarially generated blocks are colored black. Dashed arrows represent interlink pointers included in the adversary's suffix proof. Wavy lines imply one or more blocks. Firm lines imply the previousId relationship between two sequential blocks.***

In this attack the adversary uses thorny blocks to "sew" portions of the chain adopted by an honest player to her own forked chain. This remark justifies the name given to the attack.

Note that in order to make this attack successful, the adversary has to produce only a few superblocks which let her arrogate an arbitrarily large number of blocks. Thus this attack is expected to succeed with overwhelming probability.

4 PROTOCOL UPDATE

In order to eliminate the Chainsewing Attack we propose an update to the velvet NIPoPoW protocol. The core problem is that in her suffix proof the adversary is able to claim not only blocks of forked chains, which are in majority adversarially generated due to the Common Prefix property, but also arbitrarily long parts of the chain adopted by an honest player. Since thorny blocks are accepted as valid, the verifier cannot distinguish blocks that actually belong in a chain from blocks that only seem to belong in the same chain because they are pointed to via a non-smooth pointer.

4.0.1 Honest Majority Assumption. Compared to the typical setting of 1/2-bounded adversary, the protocol we propose requires stronger honest majority assumptions to be provably secure. In particular, our protocol is secure for adversary of total hashing power less than 1/2 of the upgraded honest miners, meaning less than 1/3 of the total number of miners generating blocks with interlinks. Therefore we define the Velvet Honest Majority assumption, which will be used in our security proof.

Definition 5. Velvet Honest Majority *Let n_h be the number of upgraded honest miners. Then t out of total n parties are corrupted such that* $\frac{t}{n_h} < \frac{1-\delta}{2}$.

We describe a protocol patch that operates as follows. The NIPoPoW protocol under velvet fork works as usual but each miner constructs smooth blocks. This means that a block's interlink is constructed excluding thorny blocks. In this way, although thorny blocks are accepted in the chain, they are not taken into consideration when updating the interlink structure for the next block to be mined. No honest block could now point to a thorny superblock that may act as the passing point to the fork chain in an adversarial suffix proof. Thus, after this protocol update the adversary is only able to inject adversarially generated blocks from an honestly adopted chain to her own fork. At the same time, thorny blocks cannot participate in an honestly generated suffix proof except for some blocks in the proof's suffix (χ). This arguments holds because thorny blocks do not form a valid chain along with honestly mined blocks anymore. Consequently, as far as the blocks included in a suffix proof is concerned, we can think of thorny blocks as belonging in the adversary's fork chain for the π part of the proof, which is the comparing part between proofs. Figure 5 illustrates this remark.

The protocol patch we suggest can be summarized as follows:

Protocol Patch for NIPoPows suffix proofs under velvet fork. In order to make NIPoPoWs safe under velvet fork conditions we suggest:

(1) Strengthen the Honest Majority Assumption so that $t < \frac{1-\delta}{2} n_h$, where n_h is the number of upgraded honest players.
(2) The NIPoPoW protocol under velvet fork works as usual but a miner constructs a block's interlink without pointers to thorny blocks.

The following Lemmas come as immediate results from the suggested protocol update.

Lemma 4.1. *A velvet suffix proof constructed by an honest player cannot contain any thorny block.*

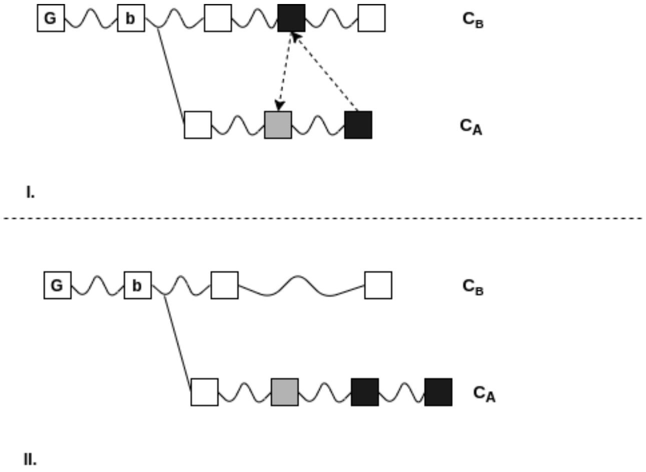

Figure 5: *The adversarial fork chain C_A and chain C_B of an honest player. Thorny blocks are colored black. Dashed arrows represent interlink pointers. Wavy lines imply one or more blocks. When an adversarially generated block is sewed from C_B into the adversary's suffix proof the verifier conceives C_A as longer and C_B as shorter. I: The real picture of the chains. II: Equivalent picture from the verifier's perspective considering the blocks included in the corresponding suffix proof for each chain.*

Lemma 4.2. *Let $\mathcal{P}_{\mathcal{A}} = (\pi_{\mathcal{A}}, \chi_{\mathcal{A}})$ be a velvet suffix proof constructed by the adversary and block b_s, generated at round r_s, be the most recent smooth block in the proof. Then $\forall r : r < r_s$ no thorny blocks generated at round r can be included in $\mathcal{P}_{\mathcal{A}}$.*

Proof. By contradiction. Let b_t be a thorny block generated at some round $r_t < r_s$. Suppose for contradiction that b_t is included in the proof. Then, because $\mathcal{P}_{\mathcal{A}}$ is a valid chain as for interlink pointers, there exist a block path made by interlink pointers starting from b_s and resulting to b_t. Let b' be the most recently generated thorny block after b_t and before b_s included in $\mathcal{P}_{\mathcal{A}}$. Then b' has been generated at a round r' such that $r_t \leq r' < r_s$. Then the block right after block b' in $\mathcal{P}_{\mathcal{A}}$ must be a thorny block since it points to b' which is thorny. But b' is the most recent thorny block after b_t, thus we have reached a contradiction.

□

Lemma 4.3. *Let $\mathcal{P}_{\mathcal{A}} = (\pi_{\mathcal{A}}, \chi_{\mathcal{A}})$ be a velvet suffix proof constructed by the adversary. Let b_t be the oldest thorny bock included in $\mathcal{P}_{\mathcal{A}}$ which is generated at round r_t. Then any block $b = \{b : b \in \mathcal{P}_{\mathcal{A}} \wedge b$ generated at $r \geq r_t\}$ is thorny.*

Proof. By contradiction. Suppose for contradiciton that b_s is a smooth block generated at round $r_s > r_t$. Then from Lemma 4.2 any block generated at round $r < r_s$ is smooth. But b_t is generated at round $r_t < r_s$ and is thorny, thus we have reached a contradiction.

□

The following corollary emerges immediately from Lemmas 4.2, 4.3. This result is illustrated in Figure 6.

Corollary 4.4. *Any adversarial proof $\mathcal{P}_{\mathcal{A}} = (\pi_{\mathcal{A}}, \chi_{\mathcal{A}})$ containing both smooth and thorny blocks consists of a prefix smooth subchain followed by a suffix thorny subchain.*

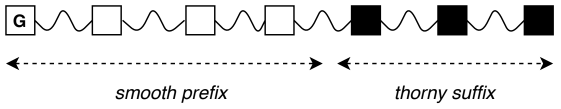

Figure 6: *In the general case the adversarial velvet suffix proof $\mathcal{P}_{\mathcal{A}} = (\pi_{\mathcal{A}}, \chi_{\mathcal{A}})$ consists of an initial part of smooth blocks followed by thorny blocks.*

We now describe the algorithms needed by the upgraded miner, prover and verifier. The upgraded miner acts as usual except for including the interlink of the newborn block in the coinbase transaction. In order to construct an interlink containing only the smooth blocks, the miner keeps a copy of the "smooth chain" (C_S) which consists of the smooth blocks existing in the original chain C. The algorithm for extracting the smooth chain out of C is given in Algorithm 1. Function *isSmoothBlock(B)* checks whether a block B is a smooth velvet by calling *isSmoothPointer(B,p)* for every pointer p in B's interlink. Function *isSmoothPointer(B,p)* returns *true* if p is a valid pointer, in essence a pointer to the most recent *smooth velvet* for the level denoted by the pointer itself. The *updateInterlink* algorithm is given in Algorithm 2, which is essentially the same as in the case of a hard/soft fork, except for working on the smooth chain C_S instead of C.

The construction of the velvet suffix prover is given in Algorithm 3, which is essentially the same to that of a hard/soft fork except for working on smooth chain C_S instead of C.

In conclusion the Verify algorithm for the NIPoPoW suffix protocol remains the same as in the case of hard or soft fork.

After these changes the honest prover cannot contain any thorny blokcs in his suffix NIPoPow even if these blocks are part of C_B. Therefore the adversary could try to supress honestly generated blocks in C_B in order to reduce the blocks that can represent the honest chain in a proof. In parallel, while the adversary mines suppressive thorny blocks on C_B she can still use her blocks in her NIPoPoW proofs, by chainsewing them. Consequently, even if a suppression attempt does not succeed, in case for example that a second honestly generated block is soon enough published, she does not drop the thorny block she generated but include it in her proof.

More in detail, consider that the adversary wishes to attack a specific block level μ_B and generate a NIPoPow proof containing a block b which contains a double spending transaction. Then she acts as follows. She may mine on her fork chain $C_{\mathcal{A}}$ but when she observes a μ_B-level block in C_B she mines a thorny block on C_B which jumps onto her fork chain, in order to suppress this μ_B block. If the suppression succeeds she has managed to damage the μ_B superchain and mine a block that she can afterwards use in her proof. If the suppression does not succeed she can still use the thorny in her proof. The above are illustrated in Figure 7.

Algorithm 1: Compute smooth chain

```
function smoothChain($C$):
    $C_S = \{\mathcal{G}\}$
    $k \leftarrow 1$
    while $C[-k] \neq \mathcal{G}$ do
        if isSmoothBlock($C[-k]$) then
            $C_S \leftarrow C_S \cup C[-k]$
        end
        $k \leftarrow k + 1$
    end
    return $C_S$
end function

function isSmoothBlock(B):
    if $B = \mathcal{G}$ then
        return true
    end
    for $p \in$ B.interlink do
        if $\neg$isSmoothPointer(B, p) then
            return false
        end
    end
    return true
end function

function isSmoothPointer(B, p):
    $b \leftarrow Block(B.prevId)$
    while $b \neq p$ do
        if $level(b) \geq level(p) \wedge$ isSmoothBlock(b) then
            return false
        end
        if $b = \mathcal{G}$ then
            return false
        end
        $b \leftarrow Block(b.prevId)$
    end
    return isSmoothBlock(b)
end function
```

Algorithm 2: Velvet updateInterlink

```
function updateInterlinkVelvet($C_S$):
    B' $\leftarrow C_S[-1]$
    interlink $\leftarrow$ B'.interlink
    for $\mu = 0$ to level(B') do
        interlink[$\mu$] $\leftarrow id(B')$
    end
    return interlink
end function
```

5 SECURITY PROOF

Theorem 1. ***Suffix Proofs Security under velvet fork*** *Assuming honest majority under velvet fork conditions (5) such that $t \leq (1-\delta)\frac{n_h}{2}$ where n_h the number of upgraded honest players, the non-interactive proofs-of-proof-of-work construction for computable*

Algorithm 3: Velvet Suffix Prover

```
function $ProveVelvet_{m,k}(C_S)$:
    $B \leftarrow C_S[0]$
    for $\mu = |C_S[-k].interlink|$ down to $0$ do
        $\alpha \leftarrow C_S[: -k]\{B :\} \uparrow^{\mu}$
        $\pi \leftarrow \pi \cup \alpha$
        $B \leftarrow \alpha[-m]$
    end
    $\chi \leftarrow C_S[-k :]$
    return $\pi\chi$
end function
```

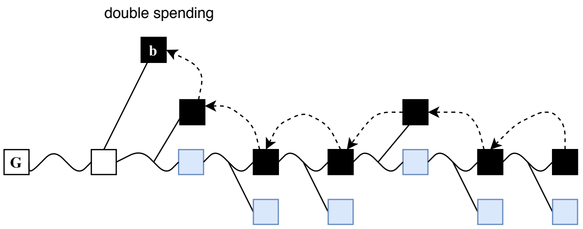

Figure 7: ***The adversary suppress honestly generated blocks and chainsew thorny blocks in C_B. Blue blocks are honestly generated blocks of a specific level of attack. The adversary tries to suppress them. If the suppression is not successful, the adversary can still use the block she mined in her proof.***

k-stable monotonic suffix-sensitive predicates under velvet fork conditions in a typical execution is secure.

Proof. By contradiction. Let Q be a k-stable monotonic suffix-sensitive chain predicate. Assume for contradiction that NIPoPoWs under velvet fork on Q is insecure. Then, during an execution at some round r_3, $Q(C)$ is defined and the verifier V disagrees with some honest participant. V communicates with adversary $\mathcal{A}$ and honest prover B. The verifier receives proofs $\pi_{\mathcal{A}}, \pi_B$ which are of valid structure. Because B is honest, π_B is a proof constructed based on underlying blockchain C_B (with $\pi_B \subseteq C_B$), which B has adopted during round r_3 at which π_B was generated. Consider $\widetilde{C}_{\mathcal{A}}$ the set of blocks defined as $\widetilde{C}_{\mathcal{A}} = \pi_{\mathcal{A}} \cup \{\bigcup\{C^r_h\{: b_{\mathcal{A}}\} : \forall b_{\mathcal{A}} \in \pi_{\mathcal{A}}, \exists h, r : b_{\mathcal{A}} \in C^r_h\}\}$ where C^r_h the chain that the honest player h has at round r.

The verifier outputs $\neg Q(C_B)$. Thus it is necessary that $\pi_{\mathcal{A}} \geq_m \pi_B$. We show that $\pi_{\mathcal{A}} \geq_m \pi_B$ is a negligible event.

Let the levels of comparison decided by the verifier be $\mu_{\mathcal{A}}$ and μ_B respectively. Let $b = LCA(\pi_{\mathcal{A}}, \pi_B)$. Let μ'_B be the adequate level of proof π_B with respect to block b. Call $\alpha_{\mathcal{A}} = \pi_{\mathcal{A}} \uparrow^{\mu_{\mathcal{A}}} \{b :\}$, $\alpha'_B = \pi_B \uparrow^{\mu'_B} \{b :\}$.

From Corollary 4.4 we have that the adversarial proof consists of a smooth interlink subchain followed by a thorny interlink subchain. We will refer to the smooth part of $\alpha_{\mathcal{A}}$ as $\alpha^{\mathcal{S}}_{\mathcal{A}}$ and to the thorny part as $\alpha^{\mathcal{T}}_{\mathcal{A}}$.

Our proof construction is based on the following intuition: we show that $\alpha_{\mathcal{A}}$ consists of three distinct parts $(\alpha^1_{\mathcal{A}}, \alpha^2_{\mathcal{A}}, \alpha^3_{\mathcal{A}})$, where $|\alpha^1_{\mathcal{A}}| = k_1, |\alpha^2_{\mathcal{A}}| = k_2$ and $|\alpha^3_{\mathcal{A}}| = k_3$. So $|\alpha_{\mathcal{A}}| = k_1 + k_2 + k_3$. Consider $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ the sets of consecutive rounds where the blocks of the three parts of $\alpha_{\mathcal{A}}$ are, respectively, generated. Consider $b_1 = LCA(\pi_{\mathcal{A}}, \pi_B)$ and $b_2 = LCA(\alpha^{\mathcal{S}}_{\mathcal{A}}, C_B)$, then part $\alpha^1_{\mathcal{A}}$ contains the blocks between b_1 and b_2. Consider b_3 the first thorny block in $\alpha_{\mathcal{A}}$, so $b_3 = \alpha^{\mathcal{T}}_{\mathcal{A}}[0]$, then $\alpha^3_{\mathcal{A}} = \alpha_{\mathcal{A}}\{b_3 :\}$. The second part $\alpha^2_{\mathcal{A}}$ contains the rest of the blocks in $\alpha_{\mathcal{A}}$.

The above are illustrated, among other, in Parts I, II of Figure 8.

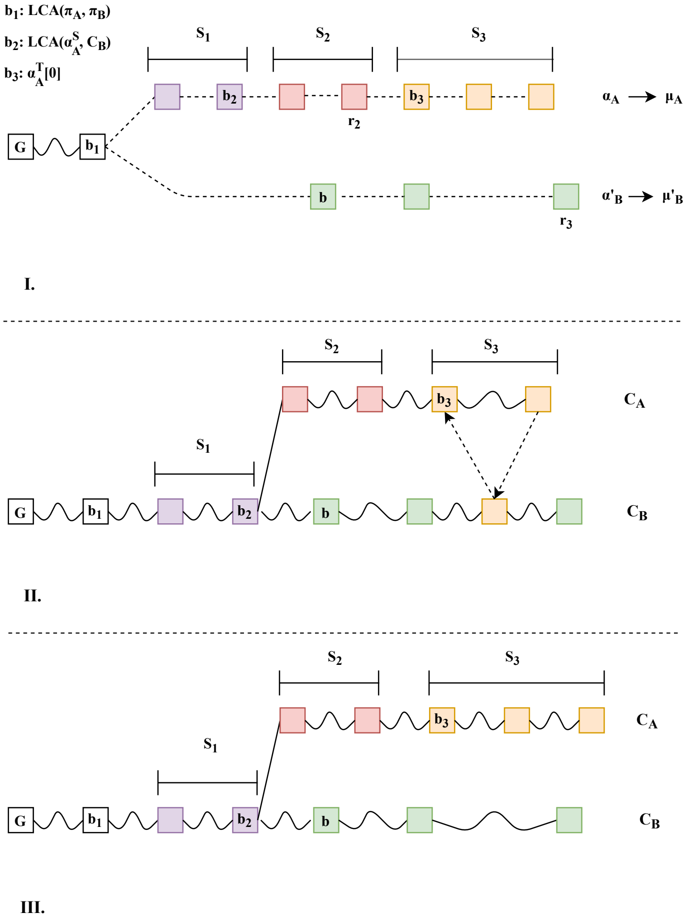

Figure 8: ***Wavy lines imply one or more blocks. Dashed lines and arrows imply interlink pointers to superblocks. I: the three round sets in two competing proofs at different levels, II: the corresponding 0-level blocks implied by the two proofs, III: blocks participating in chain C_B and block set $\widetilde{C}_{\mathcal{A}}$ from the verifier's perspective.***

We will now show three successive claims under velvet fork conditions: First, $\alpha^{\mathcal{S}}_{\mathcal{A}}$ and $\alpha'_B \downarrow$ are mostly disjoint. Second, $a_{\mathcal{A}}$ contains mostly adversarially generated blocks. And third, the adversary is able to produce this $a_{\mathcal{A}}$ with negligible probability.

Claim 1: $\alpha^{\mathcal{S}}_{\mathcal{A}}$ and $\alpha'_B \downarrow$ are mostly disjoint. First consider that there are no thorny blocks in the adversary's proof between b_1 and b_2. This means that if b_2 was generated at round r_{b_2} and $\alpha^{\mathcal{S}}_{\mathcal{A}}[-1]$

in round r, then $r \geq r_{b_2}$. For this case we show the statement considering the two possible cases for the relation of $\mu_{\mathcal{A}}, \mu'_B$.

Claim 1a: If $\mu'_B \leq \mu_A$ then they are completely disjoint. In such a case of inequality, every block in α_A would also be of lower level μ'_B. Because of the adequate level μ'_B we know that $C\{b :\} \uparrow^{\mu'_B} = \pi\{b :\} \uparrow^{\mu'_B}[1]$. Subsequently, any block in $\pi_A \uparrow^{\mu_A} \{b :\}[1 :]$ would also be included in proof α'_B, but $b = LCA(\pi_A, \pi_B)$ so there can be no succeeding block common in α_A, α'_B.

Claim 1b: If $\mu'_B > \mu_A$ then $|\alpha_A[1 :] \cap \alpha'_B \downarrow [1 :]| = k_1 \leq g(2^{\mu'_B - \mu_A})$.
Let's call b the first block in α'_B after block b_1. Suppose for contradiction that $k_1 > g(2^{\mu'_B - \mu_A})$. Since block b of level μ'_B is also of level μ_A, the adversary could include it in the proof but b cannot exist in both α_A, α'_B since $\alpha_A \cap \alpha'_B = \emptyset$ by definition. In case that the adversary chooses not to include b in the proof then she can include no other blocks of C_B in her proof, since it would not consist a valid chain. Therefore, the adversary can include at most the $\mu_{\mathcal{A}}$ upgraded blocks between b_1, b, which are expected to be equal to $g(2^{\mu'_B - \mu_A})$

Now consider the case where the adversary includes a thorny block $b_t = \alpha^{\mathcal{T}}_{\mathcal{A}}[0]$ after b and before b_2, thus the inequality still holds and because of Lemma 4.3 no more honestly generated blocks can be included in $\alpha_{\mathcal{A}}$ after b_t and we can immediately proceed to Claim 3 of this proof.

We conclude that $|\alpha^{\mathcal{S}}_{\mathcal{A}} \cap \alpha'_B \downarrow [1 :]| = k_1 \leq g(2^{\mu'_B - \mu_{\mathcal{A}}})$, where g the velvet parameter denoting the percentage of upgraded honest parties.

We conclude that there are at least $|\alpha_{\mathcal{A}}| - k_1$ blocks after block b in $\alpha_{\mathcal{A}}$ which are not honestly generated blocks existing in C_B. In other words, there are $|\alpha_{\mathcal{A}}| - k_1$ blocks after block b in $\alpha_{\mathcal{A}}$, which are either thorny blocks existing in C_B either don't belong in C_B.

Claim 2. At least k_3 superblocks of $\alpha_{\mathcal{A}}$ are adversarially generated. Consider that the block following b_2 in $\alpha_{\mathcal{A}}$ is a smooth one. Let's call this block b'_2. This means that b'_2 does not belong to C_B but to a fork chain. In this case round set $\mathcal{S}_2$ refers to the consecutive rounds from block b_2 and until the Common Prefix is established at C_B for that fork point. Consider $k_{2\downarrow}$ blocks in C_B are generated during $\mathcal{S}_2$. Then, bacause of the Common Prefix property on parameter $k_{2\downarrow}$, $\alpha_{\mathcal{A}}[k_1 + k_2 :]$ could contain no honestly generated blocks, so $k_2 \leq k$.

In the case that b'_2 is a thorny block, then because of Corollary 4.4 no more smooth block are included in $\alpha_{\mathcal{A}}$. So we consider that $k_2 = 0$ and can proceed to part $\alpha^3_{\mathcal{A}}$ and to Claim 3 of this proof.

Claim 3. The adversary may submit a suffix proof such that $\alpha_{\mathcal{A}} \geq \alpha_B$ with negligible probability. As argued earlier the last k_3 blocks included in $\alpha_{\mathcal{A}}$ are all thorny blocks. In the worst case all k_3 blocks are sewed from C_B. This is the worst case scenario since each adversarially generated block in C_B may have dropped one smooth block out of the chain because of selfish mining. Considering this scenario, because of the strengthened Honest Majority Assumption for (1/3)-bounded adversary, Theorem 3 for Chain Quality guarantees that the majority of the blocks in C_B was computed by honest parties, thus the honestly generated blocks in C_B for the same round set sum to more amount of hashing power.
From all the above Claims we have that:
In the first round set, because of the common underlying chain:

$$2^{\mu_{\mathcal{A}}}|\alpha^{k_1}_{\mathcal{A}}| \leq 2^{\mu'_B}|\alpha'^{k_1}_B| \tag{1}$$

Because of the adoption by an honest party of chain C_B at a later round r_3, we have for the second round set:

$$2^{\mu_{\mathcal{A}}}|\alpha^{k_2}_{\mathcal{A}}| \leq 2^{\mu'_B}|\alpha'^{k_2}_B| \tag{2}$$

In the third round set, because of good Chain Quality under the strengthened Honest Majority Assumption and Theorem 3 we have:

$$2^{\mu_{\mathcal{A}}}|\alpha^{k_3}_{\mathcal{A}}| < 2^{\mu'_B}|\alpha'^{k_3}_B| \tag{3}$$

Consequently we have:

$$2^{\mu_{\mathcal{A}}}(|\alpha^{k_1}_{\mathcal{A}}| + |\alpha^{k_2}_{\mathcal{A}}| + |\alpha^{k_3}_{\mathcal{A}}|) < 2^{\mu'_B}(|\alpha'^{k_1}_B| + |\alpha'^{k_2}_B| + |\alpha'^{k_3}_B|) \Rightarrow$$

$$2^{\mu_{\mathcal{A}}}|\alpha_{\mathcal{A}}| < 2^{\mu'_B}|\alpha'_B| \tag{4}$$

Therefore we have proven that $2^{\mu'_B}|\pi_B \uparrow^{\mu'_B}| > 2^{\mu_{\mathcal{A}}}|\pi^{\mu_{\mathcal{A}}}_{\mathcal{A}}|$. From the definition of μ_B, we know that $2^{\mu_B}|\pi_B \uparrow^{\mu_B}| > 2^{\mu'_B}|\pi_B \uparrow^{\mu'_B}|$ because it was chosen μ_B as level of comparison by the Verifier. So we conclude that $2^{\mu_B}|\pi_B \uparrow^{\mu_B}| > 2^{\mu_{\mathcal{A}}}|\pi_{\mathcal{A}} \uparrow^{\mu_{\mathcal{A}}}|$.

□

REFERENCES

[1] Kiayias A., Miller A., and Zindros D. 2017. Non-interactive proofs of proof-of-work. *IACR Cryptology ePrint Archive* (2017). https://eprint.iacr.org/2017/963.pdf

[2] Zamyatin A., Stifter N., Judmayer A., Schindler P., Weippl E., and Knottenbelt W.J. 2019. A Wild Velvet Fork Appears! Inclusive Blockchain Protocol Changes in Practice. *Zohar A. et al. (eds) Financial Cryptography and Data Security. FC 2018* (2019). https://eprint.iacr.org/2017/963.pdf

[3] Ittay Eyal and Emin Gun Sirer. 2013. Majority is not Enough: Bitcoin Mining is Vulnerable. (2013). arXiv:cs.CR/1311.0243

[4] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The bitcoin backbone protocol: Analysis and applications. *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2015), 281–310.

[5] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system. (2009). http://www.bitcoin.org/bitcoin.pdf