# NIPoPoWs under Velvet Fork

## 1 Introduction

Since the release of Bitcoin about a decade ago, the interest in cryptocurrencies has increased tremendously, while a number of other "altcoins" have been constructed in the meantime. Given that cryptocurrencies are starting to be considered a generally accepted means of payment and are used for everyday transactions, the issue of efficiently handling cryptocurrencies by light clients, such as smartphones, has become of great importance.

In this work, we consider the problem of optimizing light clients, or "SPV clients" as described in the original Bitcoin paper[6]. As blockchains are ever growing, the main setback for efficient light client applications is the processing of data amount linear to the size of the blockchain, e.g. for synchronization purposes.

Our work is based on the construction of Non-Interactive Proofs of Proof of Work[4] that achieves SPV proofs of polylogarithmic portion of the blockchain size. The NIPoPoWs construction suggests a protocol update, that could be possibly implemented by a soft or a hard fork. Given the reluctancy of the Bitcoin community to proceed to such forks, we consider the case of a velvet fork[4][8], where it suffices only a portion of the total players to be updated.

Under this scope, our contributions come as follows:

- We revise the security proof for superblock NIPoPoWs suffix proof protocol and compute a concrete value for the security parameter $m$
- We illustrate that, contrary to claims of previous works, superlight clients designed to work in a soft fork cannot be readily plugged into a velvet fork and expected to work. We present a novel attack termed the *chain-sewing* attack which thwarts the defenses of previous proposals and allows even a minority adversary to cause catastrophic failures.
- We propose the first *backwards-compatible superlight client*. We put forth an interlinking mechanism implementable through a velvet fork. We then construct a superblock NIPoPoW protocol on top of the velvet forked chain and show it allows to build superlight clients for various statements regarding the blockchain state via both "suffix" and "infix" proofs.
- We prove our construction secure in the synchronous static difficulty model against adversaries bounded to 1/4 of the mining power of the honest upgraded nodes. As such, our protocol works even if a constant minority of miners adopts it.

## 2 Model Definition and Notation

Our analysis concerns proof-of-work cryptocurrencies and is based on the standard Backbone model[3].

### 2.1 Blockchain Preliminaries

A blockchain, or simply chain, is a timely ordered sequence of blocks. In a cryptocurrecny blockchain, like Bitcoin, a block is a proof-of-work verified set of information about a number of transactions, the previous block in the block sequence and a nonce. The proof-of-work involves a computation over a cryptographic puzzle. More specifically, it involves scanning for a value called nonce, that when included in the block the total hash of the block results to a value lower than a certain threshold.

More formally, let $G(\cdot)$, $H(\cdot)$ be cryptographic hash functions. A *block* is a triple of the form $B = \langle s, x, ctr \rangle$, where $s$ is the previous block *id*, $x$ is the transactions information and $ctr \in \mathbb{N}$, such that satisfy the predicate $validBlock^T(B)$ defined as

$$H(ctr, G(s, x)) < T \tag{1}$$

The threshold parameter $T \in \mathbb{N}$ is called the block's *difficulty level*. Throughout this work we consider a constant value for the threshold $T$, although this is not the case in a real proof-of-work blockchain.

The rightmost block is the *head* the chain and is called the *Genesis* block often denoted $G$, while the whole chain is denoted $C$. So a chain $C$ with $G = \langle s, x, ctr \rangle$ can be extended by appending a block $B = \langle s', x', ctr' \rangle$ as long as it holds that $s' = H(ctr, G(s, x))$. In effect every block is connected to the previous block in the chain by containing its hash. This is called the *prevId* relationship. Figure 1 provides a high level representation of a blockchain including the bootstrap step of the very first block in the chain, where instead of the *prevId*, arbitrary data may be included in $s$.

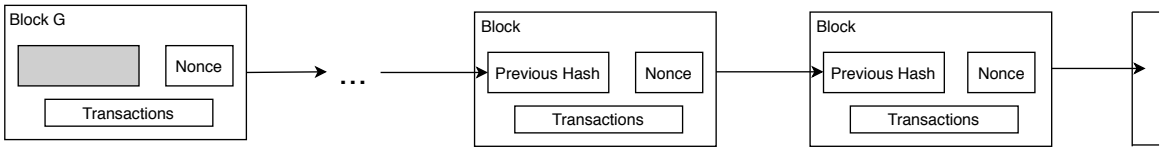


Figure 1: A high-level representation of a blockchain.

Consider a peer-to-peer network where each party may have one of the following three roles: lightweight *clients*, full *nodes* and *miners*. Miners maintain an updated copy of the chain locally, while providing computational power, also called hashpower, to extend it. In order to extend the chain by one block, the miner has to perform a proof-of-work as already described. Full nodes can be thought of as miners with zero hashpower. Full nodes are also called *provers*, since they provide proofs answering the queries for specific chain information made by lightweight clients, according to Simplified Payment Verification (SPV) described by Nakamoto[6].

According to SPV scheme lightweight clients only need to store the block headers of the longest valid chain. A block header includes only a Merkle Tree Root of the Merkle Tree comprised by the transactions included in that specific block. In order to validate that a transaction is finalized, a client needs to query the nodes until he is convinced that he has the longest valid chain, search for the block containing that transaction and finally verify an inclusion proof of the transaction in the block of interest.

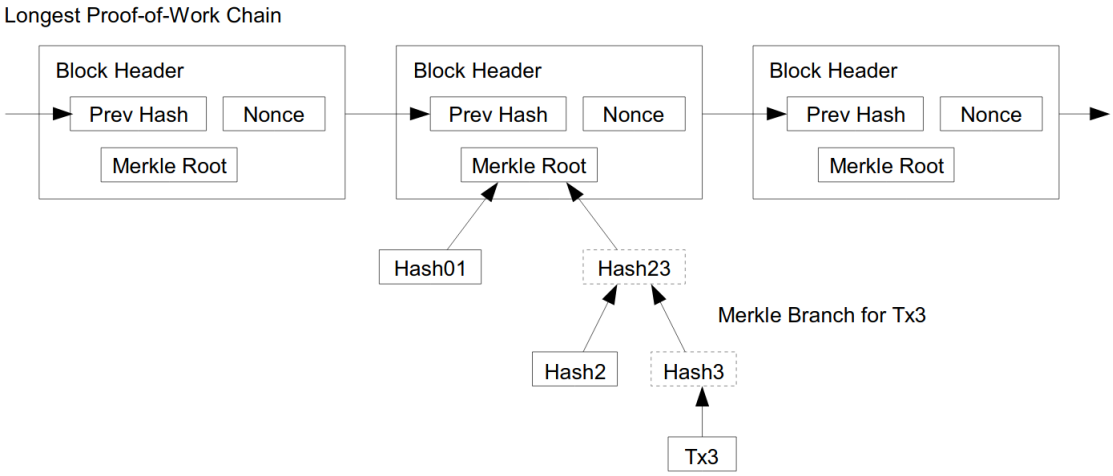


Figure 2: High level representation of blockchain data kept by a lightweight client and an inclusion proof for a transaction Tx3.[6]

In the SPV scheme a client needs to store blockchain data of linear size to the whole chain. By the time of writing Bitcoin's blockchain counts to almost 264GB and is estimated to grow more than

50GB per year. Since the growth rate of the chain is rather linear and constant, we need to construct more efficient protocols serving the needs of lightweight clients. To this end, the interaction between lightweight clients and full nodes is in our case supported by the NIPoPoWs[4] primitive which allows polylogarithmic poofs to the size of the chain.

## 2.2 The Backbone Model

The Backbone protocol is executed by an arbitrary number of parties over an unauthenticated network. We consider $n$ parties in total, $t$ of which may be controlled by an adversary.

Table 1 contains all the parameters of the Backbone protocol and will be a point of reference throughout this work.

| |
|---|
| $\lambda$ : security parameter |
| $\kappa$ : length of the hash function output |
| $n$ : number of parties mining, $t$ of which are controlled by the adversary |
| $T$ : the target hash value used by parties for solving POW |
| $t$ : number of parties controlled by the adversary |
| $\delta$ : advantage of honest parties, $\frac{t}{n-t} \leq 1-\delta$ |
| $f$ : probability at least one honest party succeeds in finding a POW in a round |
| $\epsilon$ : random variables' quality of concentration in typical executions |
| $k$ : number of (suffix) blocks for the common prefix property |
| $l$ : number of blocks for the chain quality property |
| $\mu_Q$ : chain quality parameter |
| $s$ : number of rounds for the chain growth property |
| $\tau$ : chain growth parameter |
| $L$ : the total run-tiem of the system |

Table 1: The parameters of backbone model analysis. Positive integers $n, t, L, s, l, T, k, \kappa$, positive reals $f, \epsilon, \delta, \mu_Q, \tau, \lambda$ where $f, \epsilon, \delta, \mu_Q \in (0, 1)$.

We will now give a high-level description of the Backbone Protocol and its fundamental components, namely the three supporting algorithms for *chain validation*, *chain comparison* and *proof of work*. We will also define and discuss the three properties of the protocol, namely *Common Prefix*, *Chain Quality* and *Chain Growth*. For a more formal and detailed presentation refer to the Backbone paper[3].

Consider that the protocol has already run for some rounds and a chain $C$ has been formed. Consider also an honest party that wishes to connect to the network, obtain the up-to-date version of the chain and try to extend it. The honest party connects to the network and first tries to synchronize to the current chain. The chain synchronization takes two steps to conclude. First, the newly connected peer receives a number of candidate chains by other peers in the network and validates them one by one as for the structural properties of each block *(Chain Validation)*. In particular, for each block the chain validation algorithm checks that the proof-of-work is properly solved, that the hash of the previous block is properly included in the block and that the the rest of the information included satisfies a certain validity predicate $V(\cdot)$ depending on the application. For example, in Bitcoin application it is checked that all the included transactions are valid according to the UTXO set.

Afterwards, the *Chain Comparison* algorithm is applied, where all the valid chains are compared to each other and the longest one, as for total number of blocks or total hashing power included, is considered the current active chain.

At last, in order to expand the chain by appending one more block to it, the *Proof Of Work* algorithm is applied, where the miner attempts to solve a proof of work as follows. The miner constructs the contents of the block, including the hash of the previous block and a number of new transactions published to the network. Consider that he can calculate the value $h = G(s, x)$ up to this point. Finally it remains to compute the *ctr* value so that $H(ctr, h) < T$. The protocol is

running in rounds and each party can make at most $q$ queries to function $H(\cdot)$ within a single round. If a suitable *ctr* is found, an honest party quits any queries remaining and announces the new born block to the network.

We can now define the three desired properties of the backbone protocol.

**Definition 1** (Common Prefix Property)**.** The common prefix property $Q_{cp}$ with parameter $k \in \mathbb{N}$ states that for any pair of honest players $P_1, P_2$ adopting the chains $C_1, C_2$ at rounds $r_1 \leq r_2$ respectively, it holds that $C_1^{\lceil k} \preceq C_2$.

**Definition 2** (Chain Quality Property)**.** The chain quality property $Q_{cq}$ with parameters $\mu_{cq} \in \mathbb{R}$ and $l \in \mathbb{N}$ states that for any honest party $P$ with chain $\mathcal{C}$ it holds that for any $l$ consecutive blocks of $\mathcal{C}$ the ratio of honest blocks is at least $\mu_{cq}$.

**Definition 3** (Chain Growth Property)**.** The chain growth property $Q_{cg}$ with parameters $\tau \in \mathbb{R}$ and $s \in \mathbb{N}$ states that for any honest party $P$ with chain $\mathcal{C}$, it holds that for any $s$ rounds there are at least $\tau \cdot s$ blocks added to the chain of $P$.

### 2.3 NIPoPoWs Preliminaries

### 2.4 Hard, Soft and Velvet Forks

We typically describe the two common types of a blockchain permanent fork as follows.

A *hard fork* is a consensus protocol upgrade which is not backwards compatible. This means that the changes in the protocol break the old rules since the block header's contents change. After a hard fork blocks generated by upgraded players are not accepted by the unupgraded ones. In order the protocol update to be well established, the majority of the players must be upgraded at an early point or else the non-upgraded players may maintain the longest chain under the old rules.

A *soft fork* is a consensus protocol upgrade which is backwards compatible. This is usually implemented by keeping the old rules while adding additional information in a way that unupgraded players can ignore as comments, for example, by adding adding data in the coinbase transaction. In this way unupgraded players accept blocks generated by upgraded miners as valid, while, typically, unupgraded blocks are not accepted by upgraded players. Players are motivated to upgrade in order their blocks to be accepted in the chain as valid.

A *velvet fork* is also a backwards compatible consensus protocol upgrade. Similar to soft fork additional data can be inserted in the coinbase transaction. A velvet fork requires any block compliant to the old protocol rules only to be accepted as valid by both unupgraded and upgraded players. By requiring upgraded miners to accept all blocks, even if they contain false data according to the new protocol rules, we do not modify the set of accepted blocks. Therefore, the upgrade is rather a *recommendation* and not an actual change of the consensus protocol. In reality, the blockchain is never forked. Only the codebase is upgraded and the data on the blockchain is interpreted differently[4].

The goal of this work is to provide a modified NIPoPoWs protocol so that it can be deployed under a velvet fork in a provably secure manner.

## 3 NIPoPoWs under Soft or Hard Fork

### 3.1 Suffix Proofs

NIPoPoWs suffix proofs are used to prove predicates that pertain to the suffix of the blockchain. For example, this is the case of light client synchronization to the longest valid chain. [...]

#### 3.1.1 Security of Suffix Proofs

In this section we provide the full security proof for the NIPoPoWs suffix proof protocol[4]. Apart from the proof itself (Theorem 2), we describe the definitions and lemmas being used. We try to give intuition for arguments and conclusions in each step.

Assume $t$ adversarial out of $n$ total parties, each with $q$ PoW random oracle queries per round. We define $p = \frac{T}{2^\kappa}$ the probability of a successful Random Oracle query. We will call a query to the RO *$\mu$-successful* if the RO returns a value $h$ such that $h \leq 2^{-\mu}T$.

We define the boolean random variables $X_r^\mu$, $Y_r^\mu$, $Z_r^\mu$. Fix some round $r$, query index $j$ and adversarial party index $k$ (out of $t$). If at round $r$ an honest party obtains a PoW with $id < 2^{-\mu}T$, set $X_r^\mu = 1$, otherwise $X_r^\mu = 0$. If at round $r$ exactly one honest party obtains $id < 2^{-\mu}T$, set $Y_r^\mu = 1$, otherwise $Y_r^\mu = 0$. If at round $r$ the $j$-th query of the $k$-th corrupted party is $\mu$-successful, set $Z_{rjk}^\mu = 1$, otherwise $Z_{rjk}^\mu = 0$. Let $Z_r^\mu = \sum_{k=1}^{t}\sum_{j=1}^{q} Z_{rjk}^\mu$. For a set of rounds $S$, let $X^\mu(S) = \sum_{r\in S} X_r^\mu$ and similarly define $Y_S^\mu$, $Z_S^\mu$.

**Definition 1** (Typical Execution)**.** An execution of the protocol is $(\epsilon, \eta)$-typical if:

**Block counts don't deviate.** For all $\mu \geq 0$ and any set $S$ of consecutive rounds with $|S| \geq 2^\mu \eta k$, we have:

- $(1-\epsilon)E[X^\mu(S)] < X^\mu(S) < (1+\epsilon)E[X^\mu(S)]$ and $(1-\epsilon)E[Y^\mu(S)] < Y^\mu(S)$
- $Z^\mu(S) < (1+\epsilon)E[Z^\mu(S)]$

**Round count doesn't deviate.** Let $S$ be a set of consecutive rounds such that $Z^\mu(S) \geq k$ for some security parameter $k$. Then $|S| \geq (1-\epsilon)2^\mu \frac{k}{pqt}$ with overwhelming probability.

**Chain regularity.** No insertions, no copies and no predictions [3] have occurred.

**Theorem 1** (Typicality)**.** *Executions are $(\epsilon, \eta)$-typical with overwhelming probability in $k$.*

*Proof.* **Block counts and regularity.** We refer to [3] for the full proof.

**Round count.** First, observe that for a specific round $r$ we have $Z_{rjk} \sim Bern(p)$, so for the $\mu$-level superblocks $Z_{rjk}^\mu \sim Bern(2^{-\mu}p)$ and these are jointly independent. Therefore, since for $|S|$ rounds we have $tq|S|$ adversarial RO queries, we have that $Z_S^\mu \sim \text{Bin}(tq|S|, 2^{-\mu}p)$. So $tq|S| \sim \text{NB}(Z_S^\mu, 2^{-\mu}p)$. Negative Binomial distribution is defined as $\text{NB}(r, p')$ and expresses the number of trials in a sequence of independent and identically distributed Bernoulli trials before a specified $(r)$ number of successes occurs. The expected total number of trials of a negative binomial distribution with parameters $(r, p')$ is $r/p'$. To see this, imagine an experiment simulating the negative binomial performed many times, that is a set of trials is performed until $r$ successes occur. Consider you perform $n$ experiments of total $N$ trials. Now we would expect $Np' = nr$, so $N/n = r/p'$. See that $N/n$ is just the average number of trials per experiment. So we have $E[tq|S|] = \frac{Z_S^\mu}{2^{-\mu}p} \Rightarrow E[|S|] = 2^\mu \frac{Z_S^\mu}{tqp}$. So if $Z^\mu(S) \geq k$ then $E[|S|] \geq 2^\mu \frac{k}{tqp}$. Applying a tail bound to the negative binomial distribution, we obtain that $\Pr[|S| < (1-\epsilon)E(|S|)] \in \Omega(\epsilon^2 m)$. □

**Lemma 1.** Suppose S is a set of consecutive rounds $r_1 ... r_2$ and $C_B$ is a chain adopted by an honest party at round $r_2$ of a typical execution. Let $C_S^B = \{$ b $\in C_B$ : b was generated during $S\}$. Let $\mu_A, \mu_B \in \mathbb{N}$. Suppose $C_S^B \uparrow^{\mu_B}$ is good. Suppose $C'_A$ is a $\mu_A$-superchain containing only adversarially generated blocks generated during S and suppose that $|C'_A| \geq k$. Then $2^{\mu_A}|C'_A| < 2^{\mu_B}|C_S^B \uparrow^{\mu_B}|$.

*Proof.* From $|C'_A| \geq k$ we have that $|Z_S^{\mu_A}| \geq k$. Applying Theorem 1, we conclude that $|S| \geq (1-\epsilon')2^{\mu_A}\frac{|C'_A|}{pqt}$. Applying the Chain Growth theorem [3] we obtain $|C_B^S| \geq (1-\epsilon)f|S|$. But from the goodness of $C_B^S \uparrow^{\mu_B}$, we know that $|C_B^S \uparrow^{\mu_B}| \geq (1-\delta)2^{-\mu_B}|C_B^S|$. So we have $|C_B^S \uparrow^{\mu_B}| \geq (1-\delta)2^{-\mu_B}(1-\epsilon)f|S|$ and follows that $|C_B^S \uparrow^{\mu_B}| \geq (1-\delta)2^{-\mu_B}(1-\epsilon)f(1-\epsilon')2^{\mu_A}\frac{|C'_A|}{pqt}$. Consequently we have that $2^{\mu_A}|C'_A| \leq \frac{pqt}{(1-\delta)(1-\epsilon)(1-\epsilon')f}2^{\mu_B}|C_B^S \uparrow^{\mu_B}|$.

So, according to the above equation we have that $2^{\mu_A}|C'_A| < 2^{\mu_B}|C_S^B \uparrow^{\mu_B}|$ considering that honest majority assumption holds, specifically considering that $\frac{pqt}{f} \approx \frac{t}{n-t} \leq 1$ . □

**Definition 2** (Adequate level of honest proof)**.** Let $\pi$ be an honestly generated proof constructed upon some adopted chain $C$ and let $b \in \pi$. Then $\mu'$ is defined as $\mu' = \max\{\mu : |\pi\{b :\} \uparrow^\mu| \geq$

$\max(m+1, (1-\delta)2^{-\mu}|\pi\{b:\}\uparrow^{\mu}\downarrow|)\}$. We call $\mu'$ the adequate level of proof $\pi$ with respect to block b with security parameters $\delta$ and m. Note that the adequate level of a proof is a function of both the proof $\pi$ and the chosen block b.

Intuitively, adequate is the level $\mu'$ of a proof $\pi$ for a block b if there are at least $m$ blocks after b in $\pi$ under the condition that there is good chain quality for this level, meaning that there are at least so many blocks at this level as expected considering the number of 0-level blocks.
NOTE: adequate level is mostly useful for Claim 1a of the Security Proof (Theorem 2).

**Lemma 2.** Let $\pi$ be some honest proof generated with security parameters $\delta$, m. Let C be the underlying chain, $b \in C$ be any block and $\mu'$ be the adequate level of the proof with respect to b and the same security parameters.
Then $C\{b:\}\uparrow^{\mu'} = \pi\{b:\}\uparrow^{\mu'}$.

*Proof.* $\pi\{b:\}\uparrow^{\mu'} \subseteq C\{b:\}\uparrow^{\mu'}$ is trivial. For the converse, we have that in the iteration of the *Prove for loop*[4] with $\mu = \mu^*$, the block stored in variable $B$ precedes $b$ in $C$.

Note that the Prover's for loop iterates over all levels in the interlink structure, and places in the proof all of the blocks that are of the corresponding level and succeed $B$ in $C$.

Suppose $\mu = \mu^*$ is the first *for* iteration during which the property is violated. This cannot be the first iteration since $B = C[0]$ and Genesis precedes all blocks. By induction hypothesis we see that during the iteration $\mu = mu^* + 1$, $B$ preceded $b$. From the definition of $\mu'$ we know that $\mu'$ is the highest level for which $|\pi\{b:\}\uparrow^{\mu}| \geq \max(m, (1-\delta)2^{-\mu}|\pi\{b:\}\uparrow^{\mu}\downarrow|)$.

Hence, this property cannot hold for $\mu^* > \mu$ and therefore $|\pi\{b:\}\uparrow^{\mu}| < m$ or $\neg$local-good$_{\delta}(\pi\{b:\}\uparrow\mu^*[1:], C, \mu^*)$.

In case local-good is violated, variable $B$ remains unmodified and the induction step holds. If local-good is not violated, then $|\pi\{b:\}\uparrow^{\mu^*}[1:]| < m$ and so $\pi\uparrow^{\mu^*}[-m]$, which is the updated value of $B$ at the end of $\mu^*$ iteration, precedes $b$. □

**Lemma 3.** Suppose the verifier evaluates $\pi_A \geq \pi_B$ in a protocol interaction where B is honest and assume during the comparison that the compared level of the honest party is $\mu_B$. Let $b = LCA(\pi_A, \pi_B)$ and let $\mu'_B$ be the adequate level of $\pi_B$ with respect to b. Then $\mu'_B \geq \mu_B$.

*Proof.* Because $\mu_B$ is the compared level of the honest party, from the definition of the $\geq_m$ operator, we have $2^{\mu_B}|\pi\{b:\}\uparrow^{\mu_B}| > 2^{\mu'_B}|\pi\{b:\}\uparrow^{\mu'_B}|$. This is true, otherwise the Verifier would have chosen level $\mu'_B$ as level of comparison. The proof is by contradiction. Suppose $\mu'_B < \mu_B$. By definition, $\mu'_B$ is the maximum level such that $|\pi_B\{b:\}\uparrow^{\mu}[1:]| \geq max(m, (1-\delta)2^{-\mu}|\pi_B\{b:\}\uparrow^{\mu}[1:]\downarrow|)$, therefore $\mu_B$ does not satisfy this condition. But we know that $|\pi_B\{b:\}\uparrow^{\mu}[1:]| > m$ because $\mu_B$ was selected by the Verifier. Therefore $2^{\mu_B}|\pi\{b:\}\uparrow^{\mu_B}| < (1-\delta)|C\{b:\}|$.
But also $\mu'_B$ satisfies goodness, so $2^{\mu'_B}|\pi\{b:\}\uparrow^{\mu'_B}| > (1-\delta)|C\{b:\}|$.
From the last two equations we obtain $2^{\mu_B}|\pi\{b:\}\uparrow^{\mu_B}| < 2^{\mu'_B}|\pi\{b:\}\uparrow^{\mu'_B}|$ which contradicts the initial equation. □

Intuitively the above Lemma says: the comparison level chosen by the Verifier can be no other than the adequate level in respect to block b ($LCA(\pi_A, \pi_B)$), since any other choice would be a level of non-good quality, because of the definition of the adequate level. A level of non-good quality would contain less PoW than that of the adequate level for the range of interest $C\{b:\}$.

**Theorem 2** (Security of suffix proofs)**.** *Assuming honest majority, the non-interactive proofs-of-proof-of-work construction for computable $\kappa$-stable monotonic suffix-sensitive predicates is secure with overwhelming probability in $\kappa$.*

*Proof.* By contradiction. Let $Q$ be a $\kappa-$stable monotonic suffix-sensitive chain predicate. Assume NIPoPoWs 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. Assume the execution is typical. $V$ communicates with adversary $A$ and honest prover $B$. The verifier receives proofs $\pi_A, \pi_B$. Because $B$ is honest, $\pi_B$

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 $\pi_B$ was generated. Furthermore, $\pi_A$ was generated at round $r'_3 \leq r_3$.

The verifier outputs $\neg Q(C_B)$. Thus it is necessary that $\pi_A \geq \pi_B$. We will show that this is a negligible event.

Let $b = LCA(\pi_A, \pi_B)$. Let $b^\star$ be the most recently honestly generated block in $C_B$ preceding $b$. Note that $b^\star$ necessarily exists because Genesis is honestly generated. Let the levels of comparison decided by the verifier be $\mu_A$ and $\mu_B$ respectively. Let $\mu'_B$ be the adequate level of proof $\pi_B$ with respect to block $b$. Call $\alpha_A = \pi_A \uparrow^{\mu_A} \{b :\}$, $\alpha'_B = \pi_B \uparrow^{\mu'_B} \{b :\}$.

Note that we consider the parts of the proofs succeeding block $b$ the decisive ones for the verifier's choice. This is to adversary's advantage, since the parts preceding this block demonstrate the proof-of-work contained in the common (sub)chain, thus the adversary could only include equal or less proof-of-work in her proof for this part of the chain.

We will now show three successive claims: First, $\alpha_A$ and $\alpha'_B \downarrow$ are mostly disjoint. Second, $a_A$ contains mostly adversarially generated blocks. And third, the adversary is able to produce this $a_A$ with negligible probability.

Let $\alpha_A = k_1 + k_2 + k_3$ and let $k_1, k_2, k_3$ be as defined in the following Claims.

**Claim 1:** $\alpha_A, \alpha'_B \downarrow$ are mostly disjoint. We show this by taking the two possible cases for the relation of $\mu_A$, $\mu'_B$.

*Claim 1a:* If $\mu'_B \leq \mu_A$ then they are completely disjoint. In such a case of inequality, every block in $\alpha_A$ would also be of lower level $\mu'_B$. Applying Lemma 2 to $C\{b :\} \uparrow^{\mu'_B}$ we see that $C\{b :\} \uparrow^{\mu'_B} = \pi\{b :\} \uparrow^{\mu'_B}$. Subsequently, any block in $\pi_A \uparrow^{\mu_A} \{b :\}[1 :]$ would also be included in proof $\alpha'_B$, but $b = LCA(\pi_A, \pi_B)$ so there can be no succeeding block common in $\alpha_A, \alpha'_B$.

*Claim 1b:* If $\mu'_B > \mu_A$ then $|\alpha_A[1 :] \cap \alpha'_B \downarrow [1 :]| = k_1 \leq 2^{\mu'_B - \mu_A}$.
First observe that because the adversary is winning $2^{\mu_A}|\alpha_A| > 2^{\mu'_B}|\alpha'_B| \geq 2^{\mu'_B} m \Rightarrow |\alpha_A| > 2^{\mu'_B - \mu_A} m$. Let's call $b_1$ the first block in $\alpha'_B$ after block b. Suppose for contradiction that $k_1 > 2^{\mu'_B - \mu_A}$. Since $C_B^{\mu'_B}$ is of good chain quality, this would mean that block $b_1$, of level $\mu'_B$ is also of level $\mu_A$. Since it is of level $\mu_A$ the adversary could include it in the proof but $b_1$ cannot exist in both $\alpha_A, \alpha'_B$ since $\alpha_A \cap \alpha'_B = \emptyset$ by definition. In case that the adversary chooses not to include $b_1$ in the proof then she can include no other blocks of $C_B$ in her proof, since it would not consist a valid chain.

From Claim 1$a$ and Claim 1$b$, we conclude that there are $|\alpha_A| - k_1$ blocks after block $b$ in $\alpha_A$ which do not exist in $\alpha_B \downarrow$. We now set $b_2 = LCA(C_B, \alpha_A)$. This makes $b_2$ the last block before the fork point at the 0-level chain included in the adversary's proof.

Intuition: in this case the common blocks of $\alpha_A, \alpha'_B \downarrow$ may only be blocks of level $\mu_A$ which precede the first $\mu'_B$ block appearing in $\alpha'_B$. If this block of level $\mu'_B$ was common, it could also be included in $\alpha_A$. If it is included this would be the LCA of $\alpha_A, \alpha'_B$. If it is not, then the adversary could no more include blocks from the common part of chain $C_B$ in her proof since they no longer form a valid chain in $\alpha_A$. The quantity $2^{\mu'_B - \mu_A}$ means: in the range between two consequent $\mu'_B$-level blocks, we have $n = 2^{\mu'_B}$ 0-level blocks and, thus, $2^{-\mu_A} n = 2^{\mu'_B - \mu_A}$ blocks of $\mu_A$-level.

**Claim 2:** At least $k_3$ superblocks of $\alpha_A$ are adversarially generated. We show this by showing that $\alpha_A[k_1 + k_2 + 1 :]$ contains no honestly generated blocks. Suppose for contradiction that the block $\alpha_A[i]$ for some $i \geq k_1 + k_2 + 1$ was honestly generated. This means that an honest party adopted the chain $\alpha_A[: i-1] \downarrow$ at some round $r_2 \leq r_3$. Because of the way honest parties adopt chains, the superchain $\alpha_A[: i-1]$ has an underlying properly constructed 0-level anchored chain $C_A$ such that $\alpha_A[: i-1] \subseteq C_A$. Let $j$ be the index of block $b_2$ within $\alpha_A$, $j_\downarrow$ be the index of block $b_2$ within $C_A$ and $k_{2\downarrow} = |\alpha_A[j : j + k_2] \downarrow|$. See Figure 3 for a demonstration. Observe that $|C_A[: \{\alpha_A[i-1]\}]| \geq |C_A[: j_\downarrow + k_{2\downarrow}]|$, while $C_A[j_\downarrow : j_\downarrow + k_{2\downarrow}] \not\preceq C_B$ as proved in Claim 1. But $C_A$ was adopted by an honest party at round $r_2$, which is prior to round $r_3$ during which $C_B$ was adopted by an honest party B. This contradicts the Common Prefix[3] with parameter $k_{2\downarrow}$. I t follows that with overwhelming probability in $k_{2\downarrow}$, the $k_3 = |\alpha_A| - k_2 - k_1$ last blocks of the adversarial proof

have been adversarially generated.

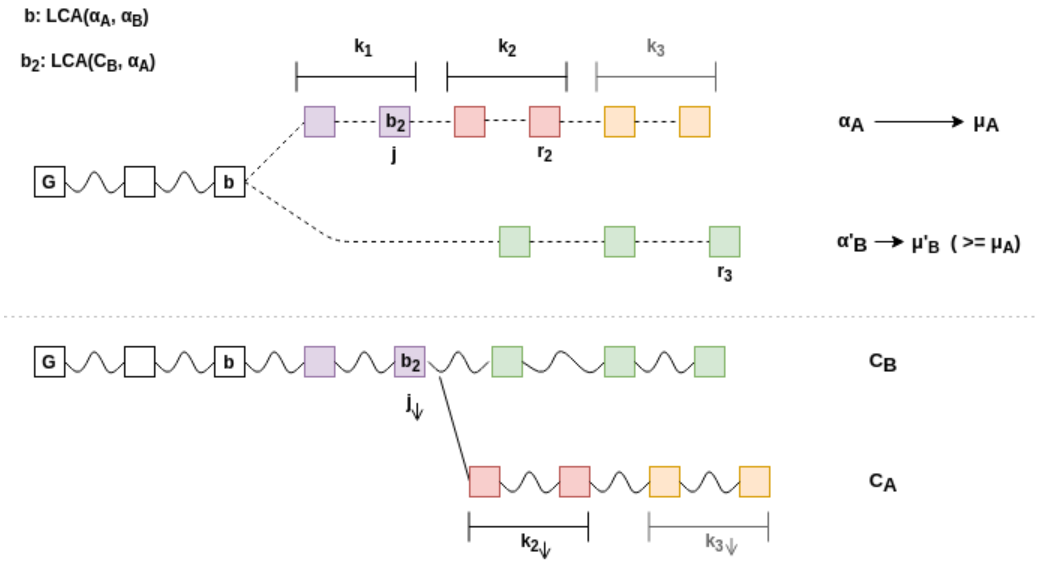


Figure 3: Two competing proofs at different levels. At the bottom the corresponding 0-level chains are represented.

Intuition: Because of Common Prefix on $k_{2\downarrow}$ parameter, where $k_{2\downarrow} = |\alpha_A[j : j + k_2] \downarrow |$, where $E[k_{2\downarrow}] = 2^{\mu_A} k_2$, there can be no honest party adopting $C_A$ at any round $i \geq k_1 + k_2 + 1$.

From these two Claims we have that $k_1$ blocks in $\alpha_A$ are blocks of the common zero-level chain, while $k_2$ blocks are blocks after the fork point at the zero-level chain. Subsequently, $k_2$ is subject to the Common Prefix $\kappa-$parameter limitations as described in the Backbone paper[3].

**Claim 3:** Adversary $A$ is able to produce $\alpha_A$ that wins against $\alpha_B$ with negligible probability.

Let $b'$ be the latest honestly generated block in $a_A$, or $b' = b^*$ if no such block exists in $a_A$. Let $r_1$ be the round when $b'$ was generated. Consider the set $S$ of consecutive rounds $r_1..r_3$. Every block in $\alpha_A[-k_3 :]$ has been adversarially generated during $S$ and $|\alpha_A[-k_3 :]| = |\alpha_A\{b' :\}| = k_3$. $C_B$ is a chain adopted by an honest party at round $r_3$ and filtering the blocks by the rounds during which they were generated to obtain $C_B^S$, we see that if $b''$ is the most recently generated block in $\alpha_B$ in a round $r \leq r_1$, then $C_B^S = C_B\{b'' :\}$. But $C_B^S \uparrow^{\mu'_B}$ is good with respect to $C_B^S$. Applying Lemma 1, we obtain that with overwhelming probability $2^{\mu_A}|\alpha_A\{b' :\}| < 2^{\mu'_B}|C_B^S \uparrow^{\mu'_B}|$, which is equal to

$$2^{\mu_A}|\alpha_A\{b' :\}| < 2^{\mu'_B}|\alpha'_B\{b'' :\}| \tag{2}$$

since $\alpha'_B$ contains all the $\mu'_B$-level blocks in $C_B^S$.

In order to complete the proof, let us now consider $\alpha_A^{k_1}$, $\alpha_A^{k_2}$, $\alpha_A^{k_3}$ the parts of $\alpha_A$ where the $k_1$, $k_2$, $k_3$ blocks reside and $\alpha_B^{k_1}$, $\alpha_B^{k_2}$, $\alpha_B^{k_3}$ the parts of $\alpha_B$ containing blocks generated in the corresponding round sets as illustrated in Figure 4.

Subsequently to the above Claims we have that:

Because of the common underlying chain in the first round set:

$$2^{\mu_A}|\alpha_A^{k_1}| \leq 2^{\mu'_B}|\alpha'^{k_1}_B| \tag{3}$$

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

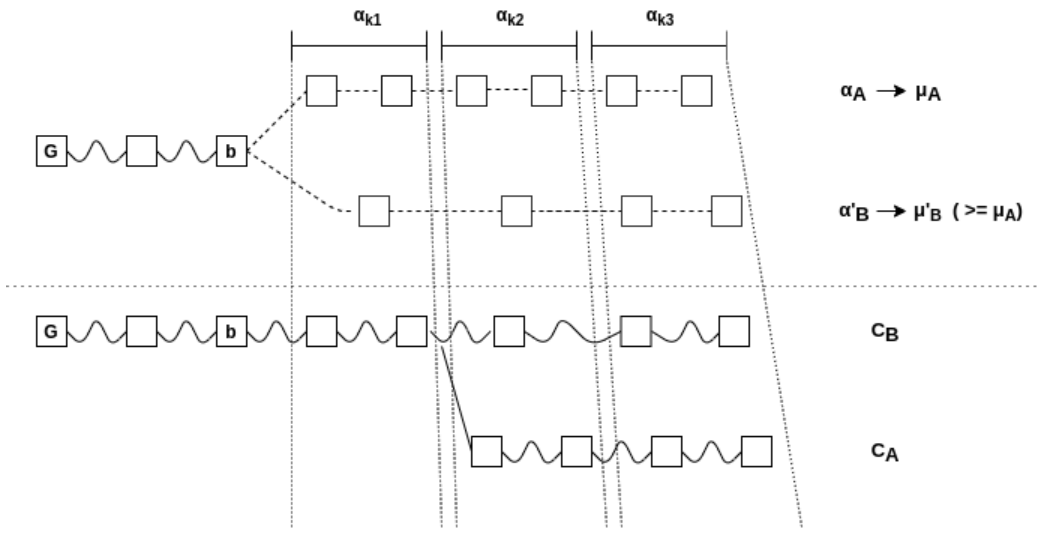


Figure 4: The three round sets in two competing proofs at different levels. The vertical dashed lines denote the area of interest, across proofs and chains, corresponding to each round set. At the bottom the corresponding 0-level chains are represented.

second round set:

$$2^{\mu_A}|\alpha_A^{k_2}| \leq 2^{\mu'_B}|\alpha'^{k_2}_B| \tag{4}$$

Because of Equation (1), we have for the third round set:

$$2^{\mu_A}|\alpha_A^{k_3}| < 2^{\mu'_B}|\alpha'^{k_3}_B| \tag{5}$$

So we have

$$2^{\mu_A}(|\alpha_A^{k_1}| + |\alpha_A^{k_2}| + |\alpha_A^{k_3}|) < 2^{\mu'_B}(|\alpha'^{k_1}_B| + |\alpha'^{k_2}_B| + |\alpha'^{k_3}_B|)$$

and finally

$$2^{\mu_A}|\alpha_A| < 2^{\mu'_B}|\alpha'_B| \tag{6}$$

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

□

It remains to calculate the security parameter $m$ that guarantee that all the above hold true in every implementation. It suffices to compute a security parameter value for each set of rounds $k_1, k_2, k_3$, so that the proof equations 3, 4, 5 hold and then sum these values to obtain parameter $m$.

In the first set of rounds, for the first $k_1$ blocks in $\alpha_A$, we only need one block included in $\alpha_B$ for the part of the proof described in Equation 3. In the second set of rounds we need $2^{-\mu_B}\kappa$ blocks for the part of the proof described in Equation 4, just as it directly results from the Common Prefix property. In order to make $m$ independent of any specific level it suffices to consider the upper bound of $\kappa$ blocks for this set of rounds. In the last set of rounds we need at least $\kappa$ adversarially generated blocks in $\alpha_A^{k_3}$ so that Lemma 1 is applicable. Since we assume honest majority, obliging to at least $\kappa$ blocks for this set of rounds suffices to guarantee for Equation 5.

So, we finally conclude to the following upper bound for the value of the security parameter:

$$m = 2\kappa + 1 \tag{7}$$

### 3.2 Infix Proofs

### 3.3 Succinctness

## 4 NIPoPoWs under Velvet Fork

### 4.1 Velvet Interlinks

More recently, velvet forks have been introduced [8]. In a velvet fork, blocks created by upgraded miners (called *velvet blocks*) are accepted by unupgraded miners as in a soft fork. Additionally, blocks created by unupgraded miners are also accepted by upgraded miners. This allows the protocol to upgrade even if only a minority of miners chooses to upgrade. To maintain backwards compatibility and to avoid causing forks, the additional data included in a block is *advisory* and must be accepted whether it exists or not. Even if the additional data is invalid or malicious, upgraded nodes (in this context also called *velvet nodes*) are forced to accept the blocks. The simplest approach to velvet fork the chain for interlinking purposes is to have upgraded miners include the interlink pointer in the blocks they produce, but accept blocks with missing or incorrect interlinks. As we show in the next section, this approach is flawed and susceptible to unexpected attacks. A surgical change in the way velvet blocks are produced is necessary to achieve proper security.

In a velvet fork, only a minority of honest parties needs to support the protocol changes. We refer to this percentage as the "velvet parameter".

**Definition 4** (Velvet Parameter)**.** The *velvet parameter* $g$ is defined as the percentage of honest parties that have upgraded to the new protocol. The absolute number of honest upgraded parties is denoted $n_h$ and it holds that $n_h = g(n - t)$.

Velvet forks maintain backwards and forwards compatibility. This requires any block produced by upgraded miners to be accepted by unupgraded nodes (as in a soft fork), but also blocks produced by unupgraded miners to be accepted by upgraded nodes. For the particular case of superblock NIPoPoWs under velvet forks, upgraded miners must include the interlink data structure within their blocks, but must also accept blocks missing the interlink structure or Velvet forks maintain backwards and forwards compatibility. This requires any block produced by upgraded miners to be accepted by unupgraded nodes (as in a soft fork), but also blocks produced by unupgraded miners to be accepted by upgraded nodes. containing an invalid interlink. Unupgraded honest nodes will produce blocks that contain no interlink, while upgraded honest nodes will produce blocks that contain truthful interlinks. Therefore, any block with invalid interlinks will be adversarially generated. However, such blocks cannot be rejected by the upgraded nodes, as that would give the adversary an opportunity to cause a hard fork.

A block generated by the adversary can thus contain arbitrary data in the interlink and yet be adopted by an honest party. Because the honest prover is an upgraded full node, it can determine what the correct interlink pointers are by examining the whole previous chain, and can thus deduce whether a block contains invalid interlink data. In that case, the prover can simply treat such blocks as unupgraded. 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.

In any velvet protocol, a specific portion within a block, which is treated as a comment by unupgraded nodes, is reused to contain auxiliary data by upgraded miners. Because these auxiliary data can be deterministically calculated, upgraded full nodes can verify the authenticity of the data in a new block they receive. We distinguish blocks based on whether they follow the velvet protocol rules or they deviate from them.

**Definition 5** (Smooth and Thorny blocks)**.** A block in a velvet protocol upgrade is called *smooth* if it contains auxiliary data and the data corresponds to the honest upgraded protocol. A block is called *thorny* if it contains auxiliary data, but the data differs from the honest upgraded protocol. A block can be neither smooth nor thorny if it does not contain auxiliary data.

In the case of velvet forks for interlink purposes, the auxiliary data consists of the Merkle Tree containing the interlink pointers to the most recent superblock ancestor at every level $\mu$.

## 4.2 A naïve velvet scheme.

In previous work [4], it was conjectured that superblock NIPoPoWs remain secure under a velvet fork. We call this scheme the *Naïve Velvet NIPoPoW* protocol, because it is not dissimilar from the NIPoPoW protocol in the soft fork case. In particular, the naïve velvet NIPoPoW protocol that was put forth works as follows. Each upgraded honest miner attempts to mine a block $b$ that includes interlink pointers in the form of a Merkle Tree included in its coinbase transaction. For each level $\mu$, the interlink contains a pointer to the most recent among all the ancestors of $b$ that have achieved at least level $\mu$, regardless of whether the referenced block is upgraded or not and regardless of whether its interlinks are valid. Unupgraded honest nodes will keep mining blocks on the chain as usual; because the status of a block as superblock does not require it to be mined by an upgraded miner, the unupgraded miners contribute mining power to the creation of superblocks as desired.

The prover in the naïve velvet NIPoPoWs then worked as follows. The honest prover constructed the NIPoPoW proof $\pi\chi$ as usual by selecting certain superblocks from his chain $\mathsf{C}$ as representatives in $\pi$ and by setting $\chi = \mathsf{C}[-k:]$. The outstanding issue in this case, however, is that these blocks in $\pi$ do not form a chain because, while superblocks, some of them may not be upgraded and they may not contain any pointers (or they may contain invalid pointers). The honest prover needs to provide a connection between two consecutive blocks $\pi[i+1]$ and $\pi[i]$ in the superchain, and suppose $\pi[i]$ is the most recent $\mu$-superblock preceding $\pi[i+1]$. The block $\pi[i+1]$ is a superblock and exists at some position $j$ in the underlying chain $\mathsf{C}$ of the prover, i.e., at $\pi[i+1] = \mathsf{C}[j]$. If $\mathsf{C}[j]$ is a smooth block, then the interlink pointer at level $\mu$ within it can be used directly. Otherwise, the prover used the *previd* pointer of $\pi[i+1] = \mathsf{C}[j]$ to repeatedly reach the parents of $\mathsf{C}[j]$, namely $\mathsf{C}[j-1], \mathsf{C}[j-2], \cdots$ until a smooth block $b$ between $\pi[i]$ and $\pi[i+1]$ was found in $\mathsf{C}$. The block $b$ then contains a pointer to $\pi[i]$, as $\pi[i]$ is also the most recent $\mu$-superblock ancestor of $b$. The blocks $\mathsf{C}[j-1], \mathsf{C}[j-2], \cdots, b$ are then included in the proof to illustrate that $\pi[i]$ is an ancestor of $\pi[i+1]$.

The argument for why the above scheme work is as follows. First of all, the scheme does not add many new blocks to the proof. In expectation, if a fully honestly generated chain is processed, after in expectation $\frac{1}{g}$ blocks have been traversed, a smooth block will be found and the connection to $\pi[i]$ will be made. Thus, the number of blocks needed in the proof increases by a factor of $\frac{1}{g}$. Security was argued as follows: An honest party includes in their proof as many blocks as in a soft forked NIPoPoW, albeit by using an indirect connection. The crucial feature is that it is not missing any superblocks. Even if the adversary creates interlinks that skip over some honest superblocks, the honest prover will not utilize these interlinks, but will use the "slow route" of level 0 instead. The adversarial prover, on the other hand, can only use honest interlinks as before, but may also use false interlinks in blocks mined by the adversary. However, these false interlinks cannot point to blocks that are of incorrect level. The reason is that the verifier can look at the hash of each block to verify its level and therefore cannot be lied to. The only problem a fake interlink can cause is that it can point to a $\mu$-superblock which is not *the most recent ancestor*, but some other block. It was then argued that the only other possibility was to point to blocks that are older $\mu$-superblock ancestors in the same chain, as illustrated in Figure 5. However, the adversarial prover can only harm herself by making use of these pointers, as the result will simply by a superchain with fewer blocks.

As such, we conclude that the honest verifier comparing the honest superchain against the adversarial superchain will reach the same conclusion in the velvet case as he would have reached in the soft fork case: Because the honest superchain in the velvet case contains the same amount of blocks as the honest superchain in the soft fork case, but the adversarial superchain in the velvet case contains fewer blocks than in the soft fork case, the comparison will remain in favor of the honest party. As we will see in the next section, this conclusion is far from straightforward.

## 4.3 The Chainsewing Attack

We now make the critical observation that a thorny block can include interlink pointers to blocks that are not its own ancestors in the 0-level chain. Because it must contain a pointer to the hash of the block it points to, they must be blocks that have been generated previously, but they may

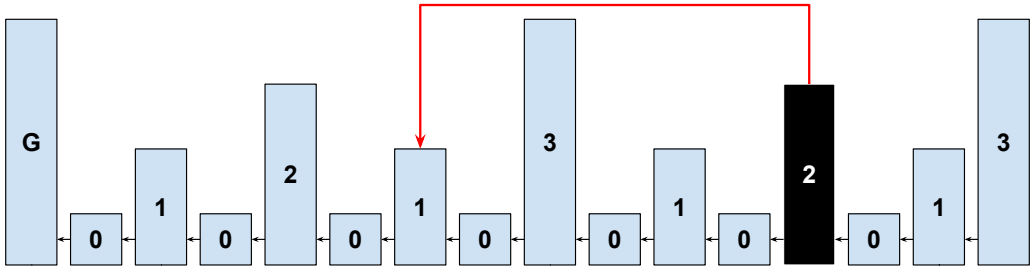


Figure 5: A thorny pointer of an adversarial block, colored black, in an honest party's chain. The thorny block points to a 1-superboock which is an ancestor 1-superblock, but not *the most recent ancestor* 1-superblock.

belong to a different 0-level chain. This is shown in Figure 6.

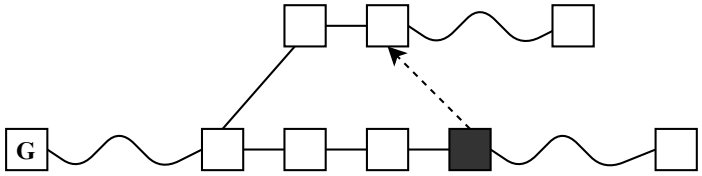


Figure 6: A thorny block, colored black, in an honest party's chain, uses its interlink to point to a fork chain.

In fact, as the interlink vector contains multiple pointers, each pointer may belong to a different fork. This is illustrated in Figure 7. The interlink pointing to arbitrary directions resembles a thorny bush.

We now present the *chain-sewing attack* against the naïve velvet NIPoPoW protocol. The attack leverages thorny blocks in order to enable the adversary to *usurp* blocks belonging to a different chain and claim it as their own. Taking advantage of thorny blocks, the adversary can produce suffix proofs containing an arbitrary number of blocks belonging to several fork chains. The attack works as follows.

Assume chain $\mathsf{C}_B$ was adopted by an honest party $B$ and chain $\mathsf{C}_{\mathcal{A}}$, a fork of $\mathsf{C}_B$ at some point, is maintained by the adversary $\mathcal{A}$. After the fork point $b = (\mathsf{C}_B \cap \mathsf{C}_{\mathcal{A}})[-1]$, the honest party produces a block extending $b$ in $\mathsf{C}_B$ containing a transaction $\mathsf{tx}$. The adversary includes a conflicting (double spending) transaction $\mathsf{tx}'$ in a block extending $b$ in $\mathsf{C}_{\mathcal{A}}$. The adversary wants to produce a suffix proof convincing a light client that $\mathsf{C}_{\mathcal{A}}$ is the longer chain. 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 party's proof, $\pi_B$, so as to achieve $\pi_{\mathcal{A}} \geq_m \pi_B$. Towards this purpose, she miners intermittently on both $\mathsf{C}_B$ and $\mathsf{C}_{\mathcal{A}}$. She produces some thorny blocks in both chains $\mathsf{C}_{\mathcal{A}}$ and $\mathsf{C}_B$ which will allow her to usurp selected blocks of $\mathsf{C}_B$ and present them to the light client as if they belonged to $\mathsf{C}_{\mathcal{A}}$ in her suffix proof.

The general form of this attack for an adversary sewing blocks to one forked chain is illustrated in Figure 8. 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, jumping between $\mathsf{C}_{\mathcal{A}}$ and $\mathsf{C}_B$, a chain of blocks crossing forks is formed, which the adversary claims as part of her suffix proof. Blocks of both chains are included in this proof and a verifier cannot distinguish the non-smooth pointers participating in this proof chain and, as a result, considers it a valid proof. Importantly, the adversary must ensure that any blocks usurped from the honest chain are not included in the honest NIPoPoW to force the NIPoPoW verifier to consider an earlier LCA block $b$; otherwise, the

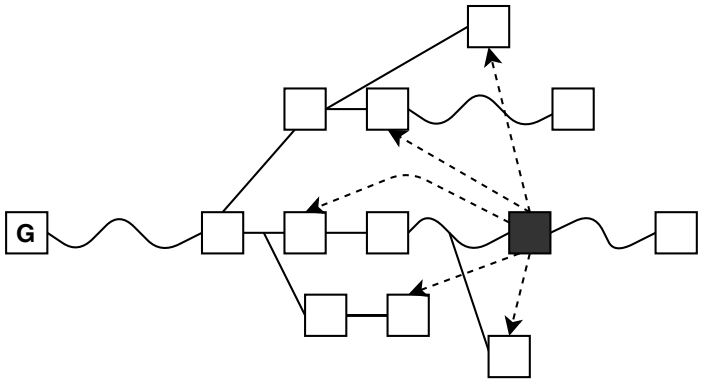


Figure 7: A thorny block appended to an honest party's chain. The dashed arrows are interlink pointers.

adversary will compete after a later fork point, negating any sewing benefits.

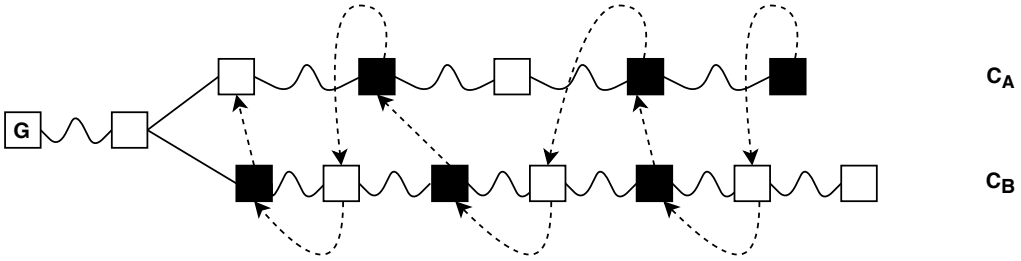


Figure 8: Generic Chainsewing Attack. $\mathsf{C}_B$ is the chain of an honest party and $\mathsf{C}_{\mathcal{A}}$ the adversary's chain. Adversarially generated blocks are colored black. Dashed arrows represent interlink pointers included in the adversary's suffix proof. Wavy lines imply one or more blocks.

This generic attack can be made concrete as follows. The adversary chooses to attack at some level $\mu_{\mathcal{A}} \in \mathbb{N}$ (ideally, if the honest verifier does not impose any succinctness limits, the adversary sets $\mu_A = 0$). As shown in Figure 9, she first generates a block $b'$ in her forked chain $\mathsf{C}_{\mathcal{A}}$ containing the double spend, and a block $a'$ in the honest chain $\mathsf{C}_B$ which thorny-points to $b'$. Block $a'$ will be accepted as valid in the honest chain $\mathsf{C}_B$ despite the invalid interlink pointers. The adversary also chooses a desired superblock level $\mu_B \in \mathbb{N}$ that she wishes the honest party to attain. Subsequently, the adversary waits for the honest party to mine and sews any blocks mined on the honest chain that are of level below $\mu_B$. However, she must bypass blocks that she thinks the honest party will include in their final NIPoPoW, which are of level $\mu_B$ (the blue block designated $c$ in Figure 9). To bypass a block, the adversary mines her own thorny block $d$ on top of the current honest tip (which could be equal to the block to be bypassed, or have progressed further), containing a thorny pointer to the block preceding the block to be bypassed and hoping that it will not exceed level $\mu_B$ (if it exceeds that level, she discards her $d$ block). Once $m$ blocks of level $\mu_B$ have been bypassed in this manner, the adversary starts bypassing blocks of level $\mu_B - 1$, because the honest NIPoPoW will start including lower-level blocks. The adversary continues descending in levels until a sufficiently low level $\min \mu_B$ has been reached at which point it becomes uneconomical for the adversary to continue bypassing blocks (typically for a 1/4 adversary, $\min \mu_B = 2$). At this point, the adversary

forks off of the last sewed honest block. This last honest block will be used as the last portion of the adversarial $\pi$ part of the NIPoPoW proof. She then independently mines a $k$-long suffix for the $\chi$ portion and creates her NIPoPoW $\pi\chi$. Lastly, she waits for enough time to pass so that the honest party's chain progresses sufficiently to make the previous bypassing guesses correct and so that no blocks in the honest NIPoPoWs coincide with blocks that have not been bypassed. This requires to wait for the following blocks to appear in the honest chain: $2m$ blocks of level $\mu_B$; after the $m^{\text{th}}$ first $\mu_B$-level block, a further $2m$ blocks of level $\mu_B - 1$; after the $m^{\text{th}}$ such block, a further $2m$ blocks of the preceding level, and so on until level 0 is reached.

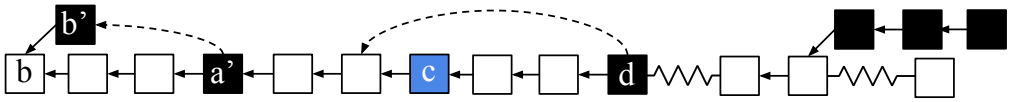


Figure 9: A portion of the concrete Chainsewing Attack. The adversary's blocks are shown in black, while the honestly generated blocks are shown in white. Block $b'$ contains a double spend, while block $a'$ sews it in place. The blue block $c$ is a block included in the honest NIPoPoW, but it is bypassed by the adversary by introducing block $d$ which, while part of the honest chain, points to $c$'s parent. After a point, the adversary forks off and creates $k = 3$ of their own blocks.

In this attack the adversary uses thorny blocks to "sew" portions of the honestly adopted chain to her own forked chain. This justifies the name given to the attack. Note that in order to make this attack successful, the adversary needs only produce few superblocks, but she can arrogate an arbitrarily large number of honestly produced blocks. Thus the attack succeeds with non-negligible probability.

#### 4.3.1 Chainsewing Attack Simulation

To measure the success rate of the chainsewing attack against the naïve NIPoPoW construction, we implemented a simulation to estimate the probability of the adversary generating a winning NIPoPoW against the honest party. Our experimental setting is as follows. We fix $\mu_A = 0$ and $\mu_B = 10$ as well as the required length of the suffix $k = 15$. We fix the adversarial mining power to $t = 1$ and $n = 5$ which gives a 20% adversary. We then vary the NIPoPoW security parameter for the $\pi$ portion from $m = 3$ to $m = 30$. We then run 100 Monte Carlo simulations and measure whether the adversary was successful in generating a competing NIPoPoW which compares favourably against the adversarial NIPoPoW.

For performance reasons, our model for the simulation slightly deviates from the Backbone model on which the theoretical analysis of Section **??** is based and instead follows the simpler model of Ren [7]. This model favours the honest parties, and so provides a lower bound for probability of adversarial success, strengthening our results. Here, block arrival is modelled as a Poisson process and blocks are deemed to belong to the adversary with probability $t/n$, while they are deemed to belong to the honest parties with probability $(n - t)/n$. Block propagation is assumed instant and every party learns about a block as soon as it is mined. As such, the honest parties are assumed to work on one common chain and the problem of non-uniquely successful rounds does not occur.

We consistently find a success rate of approximately 0.26 which remains more or less constant independent of the security parameter, as expected. We plot our results with 95% confidence intervals in Figure 10. This is in contrast with the best previously known attack in which, for all examined values of the security parameter, the probability of success remains below 1%.

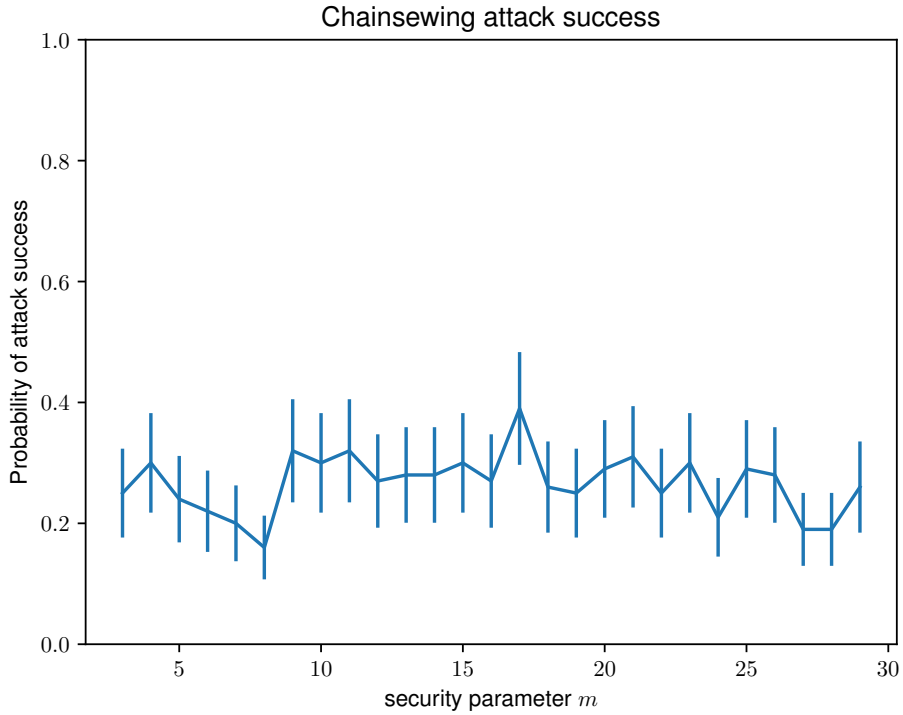


Figure 10: The measured probability of success of the Chainsewing attack mounted under our parameters for varying values of the security parameter $m$. Confidence intervals at 95%.

## 4.4 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 was able to claim not only blocks of shorter forked chains, but also arbitrarily long parts of the chain generated by an honest party. 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 from a thorny block.

The idea for a secure protocol is to distinguish the smooth from the thorny blocks, so that smooth blocks can never point to thorny blocks. In this way we can make sure that thorny blocks acting as passing points to fork chains, as block $a'$ does in Figure 9, cannot be pointed to by honestly generated blocks. Therefore, the adversary cannot utilize honest mining power to construct a stronger suffix proof for her fork chain. Our velvet construction mandates that honest miners create blocks that contain interlink pointers pointing only to previous smooth blocks. As such, newly created smooth blocks can only point to previously created smooth blocks and not thorny blocks. Following the terminology of Section **??**, the smoothness of a blocks in this new construction is a stricter notion than smoothness in the naïve construction.

In order to formally describe the suggested protocol patch, redefine the notion of a smooth block recursively by introducing the notion of a smooth interlink pointer.

**Definition 6** (Smooth Pointer)**.** A *smooth pointer* of a block $b$ for a specific level $\mu$ is the interlink pointer to the most recent $\mu$-level smooth ancestor of $b$.

We describe a protocol patch that operates as follows. The superblock NIPoPoW protocol works as usual but each honest miner constructs smooth blocks whose interlink contains only smooth pointers; thus it 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 ($\chi$). This argument 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 are concerned, we can think of thorny blocks as belonging in the adversary's fork chain for the $\pi$ part of the proof, which is the comparing part between proofs. Figure 11 illustrates this remark. The velvet NIPoPoW verifier is also modified to only follow interlink pointers, and never previd pointers (which could be pointing to thorny blocks, even if honestly generated).

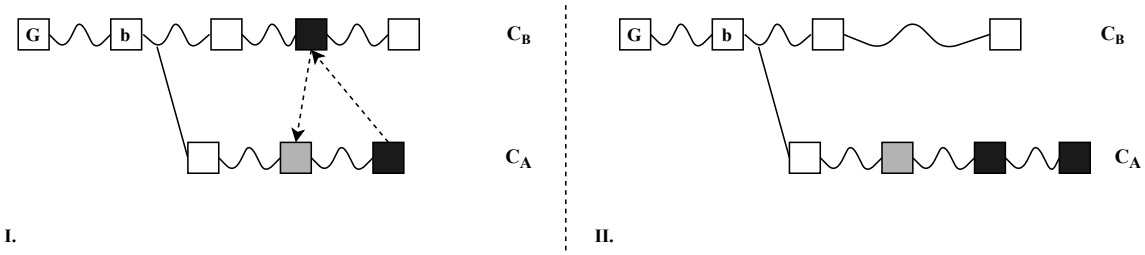


Figure 11: The adversarial fork chain $\mathsf{C}_A$ and chain $\mathsf{C}_B$ of an honest party. Thorny blocks are colored black. Dashed arrows represent interlink pointers. Wavy lines imply one or more blocks. After the protocol update, when an adversarially generated block is sewed from $\mathsf{C}_B$ into the adversary's suffix proof the verifier conceives $\mathsf{C}_A$ as longer and $\mathsf{C}_B$ 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.

With this protocol patch we conclude that the adversary cannot usurp honest mining power to her fork chain. This comes with the cost that the honest prover cannot utilize thorny blocks despite belonging in the honest chain. Due to this fact we define the Velvet Honest Majority Assumption for (1/4)-bounded adversary under which the security of the protocol is guaranteed.

**Definition 7** (Velvet Honest Majority)**.** Let $n_h$ be the number of upgraded honest miners. Then $t$ out of total $n$ parties are corrupted such that $\frac{t}{n_h} < \frac{1-\delta_v}{3}$.

The number of upgraded honest miners can be calculated via the "velvet parameter".

**Definition 3** (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 $n_h = g(n-t)$.*

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

**Lemma 4.** A velvet suffix proof constructed by an honest party cannot contain any thorny block.

**Lemma 5.** 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 6.** 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 $b = \{b : b \in \mathcal{P}_{\mathcal{A}} \wedge \text{b generated at } r \geq r_t\}$ 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 5 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 5, 6. This result is illustrated in Figure 12.

**Corollary 1.** 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.

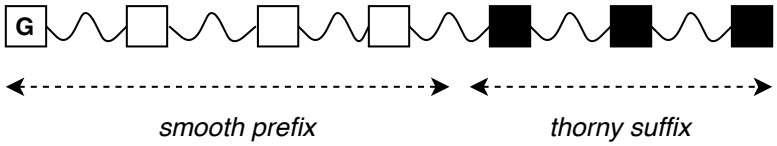


Figure 12: General case of the adversarial velvet suffix proof $\mathcal{P}_{\mathcal{A}} = (\pi_{\mathcal{A}}, \chi_{\mathcal{A}})$ consisting of an initial part of smooth blocks followed by thorny blocks.

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" ($\mathsf{C}_S$) which consists of the smooth blocks existing in the original chain $\mathsf{C}$. The algorithm for extracting the smooth chain out of $\mathsf{C}$ is given in Algorithm 1. Function *isSmoothBlock(B)* checks whether a block $B$ is smooth 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. It is the same as in the case of a soft fork, but works on the smooth chain $\mathsf{C}_S$ instead of $\mathsf{C}$.

The construction of the velvet suffix prover is given in Algorithm 3. Again it deviates from the soft fork case by working on the smooth chain $\mathsf{C}_S$ instead of $\mathsf{C}$. Lastly, the Verify algorithm for the NIPoPoW suffix protocol remains the same as in the case of hard or soft fork, keeping in mind that no *previd* links can be followed when verifying the ancestry of the chain to avoid hitting any thorny blocks.

### 4.4.1 Impossibility of a secure protocol for (1/2)-bounded adversary

During our study on the problem we failed to prove the security of several protocol constructions under $(\frac{1}{2})$-bounded adversary. We finally concluded that such a secure NIPoPoW construction is impossible under velvet fork conditions. Though it is hard to provide a typical proof for this claim, as one should consider any possible construction, we try to argue for it in this section.

**Claim:** *Assume $t$ adversarial out of $n$ total parties and $n_h = g(n-t)$ the number of upgraded honest parties. There is no construction for NIPoPoW suffix proofs under velvet fork conditions, which is both succinct and secure for every adversary, such that* $\frac{t}{n_h} < (1-\delta)$.

*Discussion.* As explained earlier, since the adversary may use the interlink structure so as to include pointers to arbitrary blocks, she may construct her own chain history utilizing the non-smooth pointers included in the thorny blocks she generates. Such an example is given in Figure 8.

Let us consider a protocol construction $p$ which allows for NIPoPoW suffix proofs under velvet fork and is secure for 1/2-bounded adversary. Our main goals are $p$ to be both secure and succinct.

**Algorithm 1** Smooth chain for suffix proofs

```
1: function smoothChain(C)
2:     C_S ← {G}
3:     k ← 1
4:     while C[−k] ≠ G do
5:         if isSmoothBlock(C[−k]) then
6:             C_S ← C_S ∪ C[−k]
7:         end if
8:         k ← k + 1
9:     end while
10:    return C_S
11: end function
12: function isSmoothBlock(B)
13:    if B = G then
14:        return true
15:    end if
16:    for p ∈ B.interlink do
17:        if ¬isSmoothPointer(B, p) then
18:            return false
19:        end if
20:    end for
21:    return true
22: end function
23: function isSmoothPointer(B, p)
24:    b ← Block(B.prevId)
25:    while b ≠ p do
26:        if level(b) ≥ level(p) ∧ isSmoothBlock(b) then
27:            return false
28:        end if
29:        if b = G then
30:            return false
31:        end if
32:        b ← Block(b.prevId)
33:    end while
34:    return isSmoothBlock(b)
35: end function
```

We can even loose our non-interactivity limitations to make it possible to challenge the submitted proofs, so that an honest player can contest against an inconsistent proof. However, note that the same power to challenge any submitted proof is also provided to the adversary.

Now assume an adversary of $\frac{1-\delta}{2} < \frac{t}{n_h} < 1-\delta$. Then, as explained in the Backbone and Selfish Mining papers [3][2] it is possible for the adversary to maintain the longest valid chain being of less than 50% chain quality in favor of adversarially generated blocks. In particular, consider that the adversary mines selfish, meaning that for every block she mines she does not immediately announce it to the network. Instead, she waits until an honest block is announced. In the worst case, consider an adversary that has a network advantage regarding the relay of her blocks. Thus in case of two competing blocks in a round the adversarial one will be appended to the chain, while the honest block is discarded in practice as a temporary fork. Because of this attack the total block ratio in the chain will result as illustrated in Figure 13. For (1/3) adversary half of the blocks in the chain are expected to be adversarial, while for (1/2) adversary no honest block is included in the chain in the worst case scenario.

Observe that the property violated in the chainsewing attack is the *prevId* relation between

**Algorithm 2** Velvet updateInterlink

```
1: function updateInterlinkVelvet(C_S)
2:     B' ← C_S[−1]
3:     interlink ← B'.interlink
4:     for μ = 0 to level(B') do
5:         interlink[μ] ← id(B')
6:     end for
7:     return interlink
8: end function
```

**Algorithm 3** Velvet Suffix Prover

```
function ProveVelvet_{m,k}(C_S)
    B ← C_S[0]
    for μ = |C_S[−k].interlink| down to 0 do
        α ← C_S[: − k]{B:}↑^μ
        π ← π ∪ α
        B ← α[−m]
    end for
    χ ← C_S[−k:]
    return πχ
end function
```

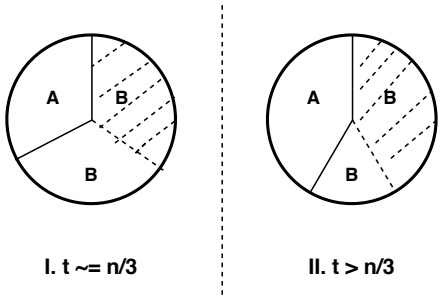


Figure 13: Pie chart of adversarially and honestly generated blocks appended in the chain during a round set S. Part **A** stands for blocks mined by the adversary while **B** for blocks mined by honest players. Lined out parts denote honestly mined blocks that were defeated by adversarially mined ones in the same round due to selfish mining.**I.** With $t = n/3$, 50% of the total blocks are adversarially generated in the worst case scenario. **II.** With $t > n/3$, more than half of the total blocks are adversarially generated in the worst case scenario.

sequential blocks. However since the main purpose of our work is to achieve succinctness, the *prevId* relations could not be utilized in a viable manner in order to contest adversarial proofs, as it would require proofs of linear length to that of the whole chain. We thus conclude that we should mostly rely on the information given by the interlink pointers as far as the validity of any proof is concerned so as to keep our protocol construction efficient.

Now assume, to honest parties' favor, that it is both possible and efficient to inspect the whole chain history that each block is commited to via its interlink pointers. Keep in mind that an adversary can keep a consistent chain considering the interlink pointers by using only her own mined blocks and, at the same time, these blocks may overwhelm the honestly generated ones. In our protocol $p$ it should be decided under what policy an honest party generates blocks and constructs suffix proofs. A decision should be made for block generation:

1. interlink data are neutral as for adversarially and honestly generated blocks

2. interlink data point out inconsistent blocks, meaning blocks with incorrect interlinks

3. interlink data exclude inconsistent blocks from being part of the valid chain formed by the super-pointers

Now let's examine the above choices.
In the first case the arguments that an honest player could provide against an inconsistent proof could only be based on the 0-level pointers. Such a contesting proof should at least provide the 0-level subchain of length equal to the following: starting from a block included in the suffix proof and is not a block of the chain until the closest block included in the suffix proof and is a valid block of the chain. This means that contesting proofs are expected to be of length $2^{\mu}$ where $\mu = log(|C|)$, and subsequently are of complexity $\mathcal{O}(|C|)$ which ruins the succinctness of our protocol.

In the second case an honest party could utilize this extra information provided in the interlink to prove an inconsistent proof wrong. However, keep in mind that the adversary could make such claims too. So, a claim of inconsistency for a specific block included should be followed by a proof of its inconsistency. Whatever the information considering the incorrect blocks in the interlink may be, the adversary could make some of it in her own generated blocks in order to contest an honest proof too. So we fall to the previous case turning to *prevId* pointers in order to prove the true chain history, where the contesting proofs are unacceptable efficiency-wise.

In the third case we demand from the honest players to validate the interlink structures of the blocks in the chain and exclude the inconsistent ones from the chain history that the interlinks provide. In this way we could possibly provide contesting proofs using information only from the interlinks thus keeping our proofs succinct. This solution would result to two different chain histories considering the interlink pointers: the one of the honest parties that includes only blocks with valid interlinks, and the one of the adversary which may form her own history of invalid blocks probably belonging to many different underlying chains but could not include honest blocks if at least one invalid block participates in the proof. This solution seems to bring us to a notable trade-off point. On the one hand we manage to uncouple from the 0-level blocks. On the other hand, we compromise that honest players cannot use valid adversarially generated blocks in their proof, while the adversary cannot include honestly generated blocks in her proofs if inconsistent blocks are also included. This solution could not work properly for (1/2)-bounded adversary though. The reason lies in the bad chain quality as pointed out earlier and shown in Figure 13. If the adversary could create his own chain history including inconsistent blocks from two different chains, then she could easily construct a suffix proof that wins over an honest one because the majority of the blocks included in the chain could be adversarial with high probability.

We conclude that such a protocol could not exist for (1/2)-bounded adversary.

### 4.4.2 Solution for (1/3)-bounded adversary

The vulnerability that makes this attack possible is the acceptance of thorny blocks. Since we operate under a velvet fork we cannot eliminate such blocks, but we need, however, to restrict the adversary from being able to claim portion of another chain as part of her own fork chain.The key observation on the Chainsewing Attack is that the adversary needs at least one adversarially generated block in an honest player's chain (block $a'$), in order to create a path of superpointers connecting blocks of two, or more, diverse chains. The final proof chain will make use of this crossing point and may contain both honest or adversarial blocks. In case the proof contains only adversarial blocks, the attack cannot harm security for an adversary of less than 1/3 of the total hashing power. This fact will be clear in the security proof section. In short, as argued in the previous section an adversary of 1/3 of the total hashing power may contribute at most 50% of the total blocks in the longest valid chain. An attacker of more than 1/3 hashing power could dominate as of total hashing power expressed in mined blocks in the chain, while the inverse would be true for an attacker of less hashing power than this threshold.

So in order to be successful, the attacker needs to also "sew" honestly generated blocks. Thus there will be at least one honest block in the superblock path which connects $a$ and $a'$, pointing to an adversarial thorny block.

The idea is to ban all blocks generated by honest players from participating in this superblock path. In this way the adversary could not misuse hashing power of the honest players and the sewed blocks could only be adversarially generated, thus the attack could never succeed for an adversary of less than 1/3 of the total hashing power.

## 4.5 Analysis of Velvet Protocol Patch

In this section, we prove the security of our scheme. Before we delve in detail into the formal details of the proof, let us first observe why the 1/4 bound is necessary through a combined attack on our construction.

After the suggested protocol update the honest prover cannot include any thorny blocks in his suffix NIPoPoW even if these blocks are part of his chain $\mathsf{C}_B$. The adversary may exploit this fact as follows. She tries to suppress high-level honestly generated blocks in $\mathsf{C}_B$, in order to reduce the blocks that can represent the honest chain in a proof. This can be done by mining a *suppressive block* on the parent of an honest superblock on the honest chain and hoping that she will be faster than the honest parties. In parallel, while she mines suppressive thorny blocks on $\mathsf{C}_B$ 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 a second honestly generated block is published soon enough, she does not lose the mining power spent but can still utilize it by including the block in her proof.

In more detail, consider the adversary who wishes to attack a specific block level $\mu_B$ and generates a NIPoPoW proof containing a block $b$ of a fork chain which contains a double spending transaction. Then she acts as follows. She mines on her fork chain $\mathsf{C}_{\mathcal{A}}$ but when she observes a $\mu_B$-level block in $\mathsf{C}_B$ she tries to mine a thorny block on $\mathsf{C}_B$ in order to suppress this $\mu_B$ block. This thorny block contains an interlink pointer which jumps onto her fork chain, but a previd pointer to the honest chain. If the suppression succeeds she has managed to damage the distribution of $\mu_B$-superblocks within the honest chain, at the same time, to mine a block that she can afterwards use in her proof. If the suppression does not succeed she can still use the thorny block in her proof. The above are illustrated in Figure 14.

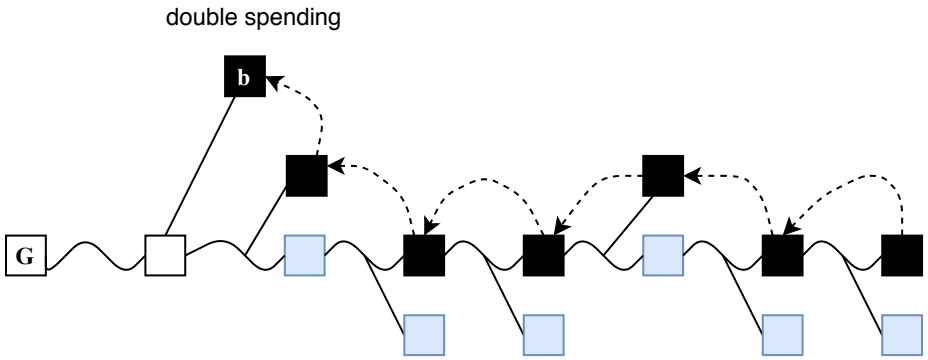


Figure 14: The adversary suppresses honestly generated blocks and chainsews thorny blocks in $\mathsf{C}_B$. Blue blocks are honestly generated blocks of some level of attack. The adversary tries to suppress them. If the suppression is not successful, the adversary can still use the block she mined in her proof.

The described attack is a combined attack which combines both superblock suppression and chainsewing. This combined attack forces us to consider the Velvet Honest Majority Assumption of (1/4)-bounded adversary, so as to guarantee that the unsuppressed blocks in $\mathsf{C}_B$ suffice for constructing winning NIPoPoW proofs against the adversarial ones.

For the analysis, we use the techniques developed in the Backbone line of work [3]. Towards that end, we follow their definitions and call a round *successful* if at least one honest party made a

successful random oracle query during the round, i.e., a query $b$ such that $H(b) \leq T$. A round in which exactly one honest party made a successful query is called *uniquely successful* (the adversary could have also made successful queries during a uniquely successful round). Let $X_r \in \{0,1\}$ and $Y_r \in \{0,1\}$ denote the indicator random variables signifying that $r$ was a successful or uniquely successful round respectively, and let $Z_r \in \mathbb{N}$ be the random variable counting the number of successful queries of the adversary during round $r$. For a set of consecutive rounds $U$, we define $Y(U) = \sum_{r \in U} Y_r$ and similarly define $X$ and $Z$. We denote $f = \mathbb{E}[X_r] < 0.3$ the probability that a round is successful.

Let $\lambda$ denote the security parameter (the output size $\kappa$ of the random oracle is taken to be some polynomial of $\lambda$). We make use of the following known [3] results. It holds that $pq(n-t) < \frac{f}{1-f}$. For the Common Prefix parameter, it holds that $k \geq 2\lambda f$. Additionally, for any set of consecutive rounds $U$, it holds that $\mathbb{E}[Z(U)] < \frac{t}{n-t} \cdot \frac{f}{1-f}|U|$, $\mathbb{E}[X(U)] < pq(n-t)|U|$, $\mathbb{E}[Y(U)] > f(1-f)|U|$. An execution is called *typical* if the random variables $X, Y, Z$ do not deviate significantly (more than some error term $\epsilon < 0.3$) from their expectations. It is known that executions are typical with overwhelming probability in $\lambda$. Typicality ensures that for any set of consecutive rounds $U$ with $|U| > \lambda$ it holds that $Z(U) < \mathbb{E}[Z(U)] - \epsilon\,\mathbb{E}[X(U)]$ and $Y(U) > (1-\epsilon)\,\mathbb{E}[Y(U)]$. From the above we can conclude to $Y(U) > (1-\epsilon)f(1-f)|U|$ and $Z(U) < \frac{t}{n-t} \cdot \frac{f}{1-f}|U| + \epsilon f|U|$ which will be used in our proofs. We consider $f < \frac{1}{20}$ a typical bound for parameter $f$. This is because in our (1/4)-bounded adversary assumption we need to reach about 75% of the network, which requires about 20 seconds [1]. Considering also that in Bitcoin the block generation time is in expectation 600 seconds, we conclude to an estimate $f = \frac{18}{600}$ or $f = 0.03$.

The following definition and lemma are known [9] results and will allow us to argue that some smooth superblocks will survive in all honestly adopted chains. With foresight, we remark that we will take $Q$ to be the property of a block being both smooth and having attained some superblock level $\mu \in \mathbb{N}$.

**Definition 4** ($Q$-block)**.** A *block property* is a predicate $Q$ defined on a hash output $h \in \{0,1\}^\kappa$. Given a block property $Q$, a valid block with hash $h$ is called a $Q$-block if $Q(h)$ holds.

**Lemma 7** (Unsuppressibility)**.** Consider a collection of polynomially many block properties $\mathcal{Q}$. In a typical execution every set of consecutive rounds $U$ has a subset $S$ of uniquely successful rounds such that

- $|S| \geq Y(U) - 2Z(U) - 2\lambda f(\frac{t}{n-t} \cdot \frac{1}{1-f} + \epsilon)$
- for any $Q \in \mathcal{Q}$, $Q$-blocks generated during $S$ follow the distribution as in an unsuppressed chain
- after the last round in $S$ the blocks corresponding to $S$ belong to the chain of any honest party.

We now apply the above lemma to our velvet construction. The following results lie at the heart of our security proof and allows us to formally argue that an honestly adopted chain will have a better superblock score than an adversarially generated chain.

**Lemma 8.** Consider Algorithm 2 under velvet fork with parameter $g$ and (1/4)-bounded velvet honest majority. Let $U$ be a set of consecutive rounds $r_1 \cdots r_2$ and $\mathsf{C}$ the chain of an honest party at round $r_2$ of a typical execution. Let $\mathsf{C}_U^S = \{b \in \mathsf{C} : \text{b is smooth} \wedge \text{b was generated during U}\}$. Let $\mu, \mu' \in \mathbb{N}$. Let $\mathsf{C}'$ be a $\mu'$ superchain containing only adversarial blocks generated during $U$ and suppose $|\mathsf{C}_U^S{\uparrow^\mu}| > k$. Then for any $\delta_3 \leq \frac{3\lambda f}{5}$ it holds that $2^{\mu'}|\mathsf{C}'| < 2^\mu(|\mathsf{C}_U^S{\uparrow^\mu}| + \delta_3)$.

*Proof.* From the Unsuppressibility Lemma we have that there is a set of uniquely successful rounds $S \subseteq U$, such that $|S| \geq Y(U) - 2Z(U) - \delta'$, where $\delta' = 2\lambda f(\frac{t}{n-t} \cdot \frac{1}{1-f} + \epsilon)$. We also know that

$Q$-blocks generated during $S$ are distributed as in an unsuppressed chain. Therefore considering the property $Q$ for blocks of level $\mu$ that contain smooth interlinks we have that $|\mathsf{C}_U^S\uparrow^\mu| \geq (1-\epsilon)g2^{-\mu}|S|$. We also know that for the total number of $\mu'$-blocks the adversary generated during $U$ that $|\mathsf{C}'| \leq (1+\epsilon)2^{-\mu'}Z(U)$. Then we have to show that $(1-\epsilon)g(Y(U)-2Z(U)-\delta') > (1+\epsilon)Z(U)$ or $((1+\epsilon)+2g(1-\epsilon))Z(U) < g(1-\epsilon)(Y(U)+\delta')$. But it holds that $(1+\epsilon)+2g(1-\epsilon) < 3$, therefore it suffices to show that

$$3Z(U) < g(1-\epsilon)(Y(U)+\delta') - 2^\mu\delta_3 \tag{8}$$

Substituting in Equation **??** the bounds of $X$, $Y$, $Z$ discussed above, it suffices to show that

$$3[\frac{t}{n-t}\cdot\frac{f}{1-f}|U| + \epsilon f|U|] < (1-\epsilon)g[(1-\epsilon)f(1-f)|U| - \delta'] - 2^\mu\delta_3$$

or

$$3f|U|\frac{f}{1-f}\cdot\frac{t}{n-t} + 3\epsilon f|U| < (1-\epsilon)g[(1-\epsilon)f(1-f)|U| - \delta'] - 2^\mu\delta_3$$

or

$$\frac{t}{n-t} < \frac{(1-\epsilon)g[(1-\epsilon)f(1-f)|U| - \delta'] - 3\epsilon f|U| - 2^\mu\delta_3}{3\frac{f}{1-f}|U|}$$

or

$$\frac{t}{n-t} < \frac{(1-\epsilon)g[(1-\epsilon)f(1-f) - \frac{\delta'}{|U|}] - 3\epsilon f - \frac{2^\mu\delta_3}{|U|}}{3\frac{f}{1-f}}$$

But $\epsilon(1-f) \ll 1$ thus we have to show that

$$\frac{t}{n-t} < \frac{(1-\epsilon)g[(1-\epsilon)f(1-f) - \frac{\delta'}{|U|}] - \frac{2^\mu\delta_3}{|U|}}{3\frac{f}{1-f}} - \epsilon'$$

or

$$\frac{t}{n-t} < \frac{g}{3}\cdot\frac{(1-\epsilon)^2 f(1-f) - \frac{(1-\epsilon)\delta'}{|U|} - \frac{2^\mu\delta_3}{|U|}}{\frac{f}{1-f}} - \epsilon' \tag{9}$$

In order to show Equation 9 we use $f \leq \frac{1}{20}$ which is a typical bound for our setting as discussed above. Because all blocks in $\mathsf{C}$ were generated during $U$ and $|\mathsf{C}| > k$, $|U|$ follows negative binomial distribution with probability $2^{-\mu}pq(n-t)$ and number of successes $k$. Applying a Chernoff bound we have that $|U| > (1-\epsilon)\frac{k}{2^{-\mu}pq(n-t)}$. Using the inequalities $k \geq 2\lambda f$ and $pq(n-t) < \frac{f}{1-f}$, we deduce that $|U| > (1-\epsilon)2^\mu 2\lambda(1-f)$. So we have that

$$\frac{\delta'}{|U|} < \frac{2\lambda f(\frac{t}{n-t}\frac{1}{1-f} + \epsilon)}{(1-\epsilon)2^\mu 2\lambda(1-f)}$$

or

$$\frac{\delta'}{|U|} < \frac{t}{n-t}\cdot\frac{f}{(1-\epsilon)(1-f)^2} + \epsilon < 0.01 + \epsilon$$

We also know that $\delta_3 \leq \frac{3\lambda f}{5}$, so

$$\frac{2^\mu\delta_3}{|U|} < \frac{2^\mu\frac{3\lambda f}{5}}{2^\mu 2\lambda(1-f)}$$

or

$$\frac{2^{\mu}\delta_3}{|U|} < \frac{3f}{10(1-f)} < 0.01 + \epsilon$$

By substituting the above and the typical $f$ parameter bound in Equation (9) we conclude that it suffices to show that $\frac{t}{n-t} < \frac{1-\epsilon''}{3}g$ which is equivalent to $\frac{t}{n-t} < \frac{1-\delta_v}{3}g$ for $\epsilon'' = \delta_v$, which is the (1/4) velvet honest majority assumption, so the claim is proven. □

**Lemma 9.** Consider Algorithm 2 under velvet fork with parameter $g$ and (1/4)-bounded velvet honest majority. Consider the property $Q$ for blocks of level $\mu$. Let $U$ be a set of consecutive rounds and $\mathsf{C}$ the chain of an honest party at the end of $U$ of a typical execution and $\mathsf{C}_U = \{b \in \mathsf{C} :$ b was generated during U$\}$. Suppose that no block in $\mathsf{C}_U$ is of level $\mu$.

Then $|U| \leq \delta_1$ where $\delta_1 = \dfrac{(2+\epsilon)2^{\mu} + \delta'}{(1-\epsilon)f(1-f) - 2\frac{t}{n-t}\frac{f}{1-f} - 3\epsilon f}$.

*Proof.* The statement results immediately form the Unsuppressibility Lemma. Suppose for contradiciton that $|U| > \delta_1$. Then from the Unsuppressibility Lemma we have that there is a subset of consecutive rounds $S$ of $U$ for which it holds that $|S| \geq Y(U) - 2Z(U) - \delta'$ where $\delta' = 2\lambda f(\frac{t}{n-t} \cdot \frac{1}{1-f} + \epsilon)$. By substituting $Y(U) > (1-\epsilon)f(1-f)|U|$ and $Z(U) < \frac{t}{n-t}\frac{f}{1-f} + \epsilon f|U|$ we have that $|S| > (2+\epsilon)2^{\mu}$ but $Q$-blocks generated during $S$ follow the distribution as in a chain where no suppression attacks occur. Therefore at least one block of level $\mu$ would appear in $\mathsf{C}_U$, thus we have reached a contradiction and the statement is proven. □

**Theorem 1** (Suffix Proofs Security under velvet fork)**.** *Assuming honest majority under velvet fork conditions (3) such that $t \leq (1-\delta_v)\frac{n_h}{3}$ where $n_h$ the number of upgraded honest parties, the Non-Interactive Proofs of Proof-of-Work construction for computable $k$-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 $r_3$, $Q(\mathsf{C})$ 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, $\pi_B$ is a proof constructed based on underlying blockchain $\mathsf{C}_B$ (with $\pi_B \subseteq \mathsf{C}_B$), which $B$ has adopted during round $r_3$ at which $\pi_B$ was generated. Consider $\widetilde{\mathsf{C}}_{\mathcal{A}}$ the set of blocks defined as $\widetilde{C}_{\mathcal{A}} = \pi_{\mathcal{A}} \cup \{\bigcup\{\mathsf{C}_h^r\{:b_{\mathcal{A}}\} : b_{\mathcal{A}} \in \pi_{\mathcal{A}}, \exists h, r : b_{\mathcal{A}} \in \mathsf{C}_h^r\}\}$ where $\mathsf{C}_h^r$ the chain that the honest party $h$ has at round $r$. Consider also $\mathsf{C}_B^S$ the set of smooth blocks of honest chain $\mathsf{C}_B$. We apply security parameter

$$m = 2k + \frac{2+\epsilon+\delta'}{\frac{t}{n-t}\frac{f}{1-f}[f(1-f) - \frac{2}{3}\frac{f}{1-f}]}$$

Suppose for contradiction that the verifier outputs $\neg Q(\mathsf{C}_B)$. 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 $\mu_B$ respectively. Let $b_0 = LCA(\pi_{\mathcal{A}}, \pi_B)$. Call $\alpha_{\mathcal{A}} = \pi_{\mathcal{A}}{\uparrow}^{\mu_{\mathcal{A}}}\{b_0:\}$, $\alpha_B = \pi_B{\uparrow}^{\mu_B}\{b_0:\}$.

From Corollary 1 we have that the adversarial proof consists of a smooth interlink subchain followed by a thorny interlink subchain. We refer to the smooth part of $\alpha_{\mathcal{A}}$ as $\alpha_{\mathcal{A}}^{\mathcal{S}}$ and to the thorny part as $\alpha_{\mathcal{A}}^{\mathcal{T}}$.

Our proof construction is based on the following intuition: we consider that $\alpha_{\mathcal{A}}$ consists of three distinct parts $\alpha_{\mathcal{A}}^1, \alpha_{\mathcal{A}}^2, \alpha_{\mathcal{A}}^3$ with the following properties. Block $b_0 = LCA(\pi_{\mathcal{A}}, \pi_B)$ is the fork point between $\pi_{\mathcal{A}}{\uparrow}^{\mu_{\mathcal{A}}}, \pi_B{\uparrow}^{\mu_B}$. Let block $b_1 = LCA(\alpha_{\mathcal{A}}^{\mathcal{S}}, \mathsf{C}_B^S)$ be the fork point between $\pi_{\mathcal{A}}^{\mathcal{S}}{\uparrow}^{\mu_{\mathcal{A}}}, \mathsf{C}_B$ as an honest prover could observe. Part $\alpha_{\mathcal{A}}^1$ contains the blocks between $b_0$ exclusive and $b_1$ inclusive generated during the set of consecutive rounds $\mathcal{S}_1$ and $|\alpha_{\mathcal{A}}^1| = k_1$. Consider $b_2$ the last block in $\alpha_{\mathcal{A}}$ generated by an honest party. Part $\alpha_{\mathcal{A}}^2$ contains the blocks between $b_1$ exclusive and $b_2$ inclusive

generated during the set of consecutive rounds $\mathcal{S}_2$ and $|\alpha^2_{\mathcal{A}}| = k_2$. Consider $b_3$ the next block of $b_2$ in $\alpha_{\mathcal{A}}$. Then $\alpha^3_{\mathcal{A}} = \alpha_{\mathcal{A}}[b_3{:}]$ and $|\alpha^3_{\mathcal{A}}| = k_3$ consisting of adversarial blocks generated during the set of consecutive rounds $\mathcal{S}_3$. Therefore $|\alpha_{\mathcal{A}}| = k_1 + k_2 + k_3$ and we will show that $|\alpha_{\mathcal{A}}| < |\alpha_B|$.

The above are illustrated, among other, in Parts I, II of Figure 15.

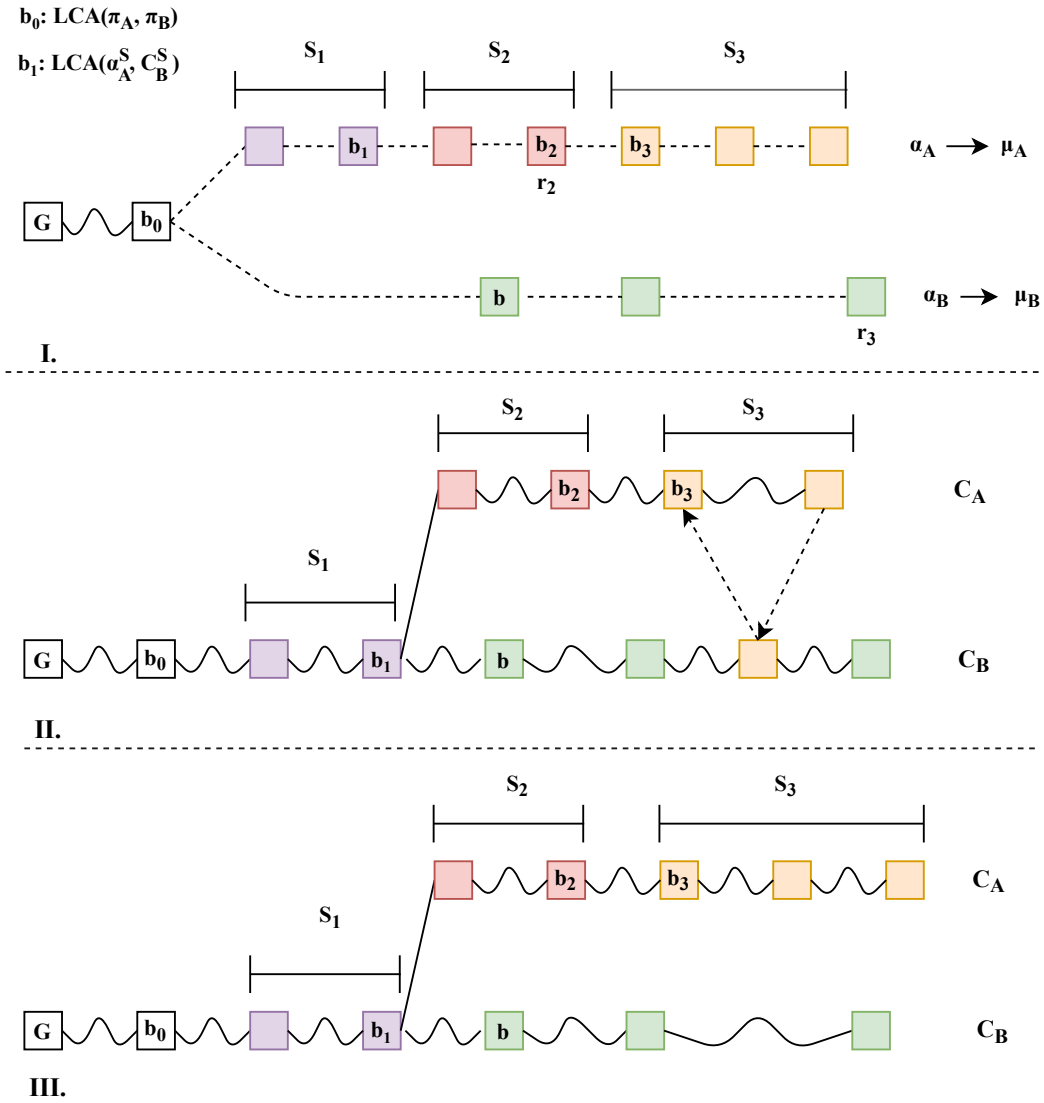


Figure 15: I. The three round sets in two competing proofs at different levels, II. the corresponding 0-level blocks implied by the two proofs, III: blocks in $\mathsf{C}_B$ and block set $\tilde{\mathsf{C}}_{\mathcal{A}}$ from the verifier's perspective.

We now show three successive claims: First that $\alpha^1_{\mathcal{A}}$ contains few blocks. Second, $\alpha^2_{\mathcal{A}}$ contains few blocks. And third, the adversary can produce a winning $a_{\mathcal{A}}$ with negligible probability.

**Claim 1:** $\alpha^1_{\mathcal{A}} = (\alpha_{\mathcal{A}}\{b_0 : b_1\} \cup b_1)$ *contains only a few blocks.* Let $|\alpha^1_{\mathcal{A}}| = k_1$. We have defined the blocks $b_0 = LCA(\pi_{\mathcal{A}}, \pi_B)$ and $b_1 = LCA(\alpha^{\mathcal{S}}_{\mathcal{A}}, \mathsf{C}^S_B)$. First observe that because of the Corollary 1 there are no thorny blocks in $\alpha^1_{\mathcal{A}}$ since $\alpha^1_{\mathcal{A}}[-1] = b_1$ is a smooth block. This means that if $b_1$ was generated at round $r_{b_1}$ and $\alpha^{\mathcal{S}}_{\mathcal{A}}[-1]$ in round $r$ then $r \geq r_{b_1}$. Therefore, $\alpha^1_{\mathcal{A}}$ contains smooth blocks of $\mathsf{C}_B$. We show the claim by considering the two possible cases for the relation of $\mu_{\mathcal{A}}, \mu_B$.

*Claim 1a: If $\mu_B \leq \mu_{\mathcal{A}}$ then $k_1 = 0$.* In order to see this, first observe that every block in $\alpha_{\mathcal{A}}$ would also be of lower level $\mu_B$. Subsequently, any block in $\alpha_{\mathcal{A}}\{b_0{:}\}$ would also be included in proof

$\alpha_B$ but this contradicts the minimality of block $b_0$.

*Claim 1b: If $\mu_B > \mu_{\mathcal{A}}$ then $k_1 \leq \dfrac{\delta_1 2^{-\mu_{\mathcal{A}}}}{(1+\epsilon)\frac{t}{n-t}\frac{f}{1-f}}$.* In order to show this we consider block $b$ the first block in $\alpha_B$. Now suppose for contradiction that $k_1 > \dfrac{\delta_1 2^{-\mu_{\mathcal{A}}}}{(1+\epsilon)\frac{t}{n-t}\frac{f}{1-f}}$. Then from lemma 9 we have that block $b$ is generated during $S_1$. But $b$ is of lower level $\mu_{\mathcal{A}}$ and $\alpha^1_{\mathcal{A}}$ contains smooth blocks of $\mathsf{C}_B$. Therefore $b$ is also included in $\alpha^1_{\mathcal{A}}$, which contradicts the minimality of block $b_0$.

Consequently, there are at least $|\alpha_{\mathcal{A}}| - k_1$ blocks in $\alpha_{\mathcal{A}}$ which are not honestly generated blocks existing in $\mathsf{C}_B$. In other words, these are blocks which are either thorny blocks existing in $\mathsf{C}_B$ either don't belong in $\mathsf{C}_B$.

**Claim 2.** *Part $\alpha^2_{\mathcal{A}} = (\alpha_{\mathcal{A}}\{b_1 : b_2\} \cup b_2)$ consists of only a few blocks.* Let $|\alpha^2_{\mathcal{A}}| = k_2$. We have defined $b_2 = \alpha^2_{\mathcal{A}}[-1]$ to be the last block generated by an honest party in $\alpha_{\mathcal{A}}$. Consequently no thorny block exists in $\alpha^2_{\mathcal{A}}$, so all blocks in this part belong in a proper zero-level chain $\mathsf{C}^2_{\mathcal{A}}$. Let $r_{b_1}$ be the round at which $b_1$ was generated. Since $b_1$ is the last block in $\alpha_{\mathcal{A}}$ which belongs in $\mathsf{C}_B$, then $\mathsf{C}^2_{\mathcal{A}}$ is a fork chain to $\mathsf{C}_B$ at some block $b'$ generated at round $r' \geq r_{b_1}$. Let $r_2$ be the round when $b_2$ was generated by an honest party. Because an honest party has adopted chain $\mathsf{C}_B$ at a later round $r_3$ when the proof $\pi_B$ is constructed and because of the Common Prefix property on parameter $k_2$, we conclude that $k_2 \leq 2^{-\mu_{\mathcal{A}}}k$.

**Claim 3.** *The adversary may submit a suffix proof such that $|\alpha_{\mathcal{A}}| \geq |\alpha_B|$ with negligible probability.* Let $|\alpha^3_{\mathcal{A}}| = k_3$. As explained earlier part $\alpha^3_{\mathcal{A}}$ consists only of adversarially generated blocks. Let $S_3$ be the set of consecutive rounds $r_2 ... r_3$. Then all $k_3$ blocks of this part of the proof are generated during $S_3$. Let $\alpha^3_B$ be the last part of the honest proof containing the interlinked $\mu_B$ superblocks generated during $S_3$. Then by applying Lemma 8 $\dfrac{m}{k}$ times we have that $2^{\mu_{\mathcal{A}}}|\alpha^3_{\mathcal{A}}| < 2^{\mu_B}(|\alpha_B^{S_3}\uparrow^{\mu_B}| + \dfrac{m\delta_3}{k})$. By substituting the values from all the above Claims and because every block of level $\mu_B$ in $a\alpha_B$ is of equal hashing power to $2^{\mu_B - \mu_{\mathcal{A}}}$ blocks of level $\mu_{\mathcal{A}}$ in the adversary's proof we have that: $2^{\mu_B}|\alpha^3_B| - 2^{\mu_{\mathcal{A}}}|\alpha^3_{\mathcal{A}}| > 2^{\mu_{\mathcal{A}}}(k_1 + k_2)$ or $2^{\mu_B}|\alpha^3_B| > 2^{\mu_{\mathcal{A}}}|\alpha^1_{\mathcal{A}} + \alpha^2_{\mathcal{A}} + \alpha^3_{\mathcal{A}}|$ or $2^{\mu_B}|\alpha_B| > 2^{\mu_{\mathcal{A}}}|\alpha_{\mathcal{A}}|$ Therefore we have proven that $2^{\mu_B}|\pi_B\uparrow^{\mu_B}| > 2^{\mu_{\mathcal{A}}}|\pi^{\mu_{\mathcal{A}}}_{\mathcal{A}}|$. □

## 4.6 Infix Proofs

The security of the original NIPoPoWs protocol suffers under velvet fork conditions for the case of infix proofs as well. Again, since blocks containing incorrect interlink pointers are accepted in the chain, the adversary may create an infix proof for a transaction included in a block mined on a different chain. This attack is presented in detail in the following.

An infix proof attack when applying the original protocol under a velvet fork should be obvious after our previous discussion. So consider the updated protocol for secure suffix proofs as described in the previous section. A problem here is that in the updated protocol some blocks are excluded from the interlink, while we should still be able to provide proofs for transactions included in any block of the chain.

For this reason, let us naively consider an additional protocol patch suggesting to include a second interlink data structure in each block, which will be updated without any block exclusion, just as described in the original protocol and will be used for constructing infix proofs only. In order to be secure we could think of allowing using pointers of the second interlink only for the *followDown* part of the algorithm. But still, the adversary may use an invalid pointer of a block visited during the *followDown* procedure and jump to a block of another chain providing a transaction inclusion proof concerning that block. This attack is illustrated in Figure 16.

Thus giving the ability to utilize invalid pointers even in a narrow block window can break the security of our protocol.

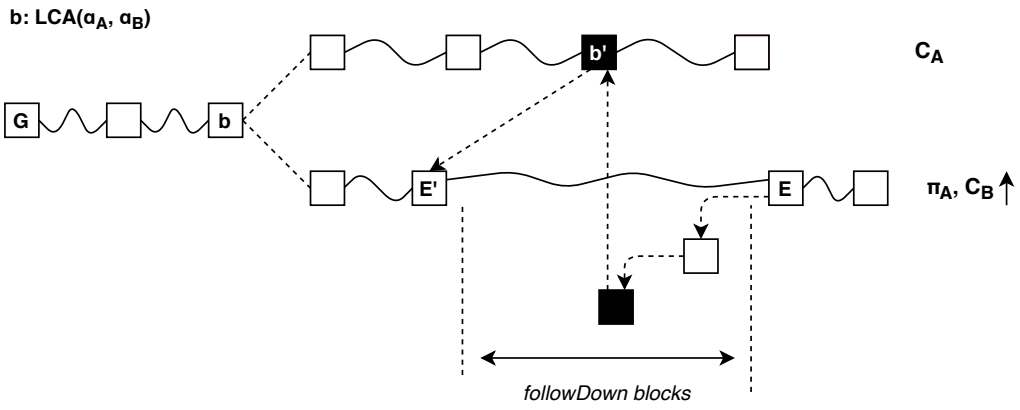


Figure 16: Adversarial fork chain $C_A$ and an adversarial infix proof based on the chain adopted by an honest player. Wavy lines imply one or more blocks. Blocks generated by the adversary are colored black. Dashed arrows represent interlink pointers included in the proof as part of the *followDown* procedure. The adversary provides infix proof for a transaction in block b'.

### Protocol patch for NIPoPoWs infix proofs under velvet fork

In order to construct secure infix proofs under velvet fork conditions, we suggest the following additional protocol patch: each upgraded miner constructs and updates an authenticated data structure for all the blocks in the chain. We suggest Merkle Mountain Ranges *(MMR)* for this structure. Now a velvet block's header additionally includes the root of this MMR.

After this additional protocol change the notion of a smooth block changes as well. Smooth blocks are now considered the blocks that contain truthful interlinks and valid MMR root too. A valid MMR root denotes the MMR that contains all the blocks in the chain of an honest full node. Note that a valid MMR contains all the blocks of the longest valid chain, meaning both smooth and thorny. An invalid MMR constructed by the adversary may contain a block of a fork chain. Consequently an upgraded prover has to maintain a local copy of this MMR locally, in order to construct correct proofs. This is crucial for the security of infix proofs, since keeping the notion of a smooth block as before would allow an adversary to produce a block $b$ in an honest party's chain, with $b$ containing a smooth interlink but invalid MMR, so she could succeed in providing an infix proof about a block of a fork chain.

### The velvet infix prover and verifier

Considering this addtional patch we can now define the final algorithms for the honest miner, infix and suffix prover, as well as for the infix verifier. Because of the new notion of smooth block, the function *isSmoothBlock()* of Algorithm 1 needs to be updated, so that the validity of the included MMR root is also checked. The updated function is given in Algorithm 4. Considering that input $\mathsf{C}_S$ is computed using Algorithm 1 with the updated *isSmoothBlock'()* function, *Velvet updateInterlink* and *Velvet Suffix Prover* algorithms remain the same as described in Algorithms 2, 3 repsectively. The velvet infix prover given in Algorithms 5, 6 respectively. In order to keep the algorithm generic enough for any infix-sensitive predicate, we provide the steps needed until the verification of the block of interest and consider that the specific predicate can be answered by a known algorithm given the block of interest. Given that the verifier is already synchronized to the longest valid chain, the infix verification algorithm only has to confirm the Merkle-Tree inclusion proof $\pi_{b'}$ for the block of interest $b'$.

Details about the construction and verification of an MMR and the respective inclusion proofs can be found in [5]. Note that equivalent solution could be formed by using any authenticated data structure that provides inclusion proofs of size logarithmic to the length of the chain. We suggest

MMRs because of they come with efficient update operations.

**Algorithm 4** Function isSmoothBlock'() for infix proof support

```
1: function isSmoothBlock'(B)
2:     if B = G then
3:         return true
4:     end if
5:     for p ∈ B.interlink do
6:         if ¬isSmoothPointer(B, p) then
7:             return false
8:         end if
9:     end for
10:    return containsValidMMR(B)
11: end function
```

**Algorithm 5** Velvet Infix Prover

```
1: function ProveInfixVelvet(C_S, b)
2:     (π, χ) ← ProveVelvet(C_S)
3:     tip ← π[−1]
4:     π_b ← MMRinclusionProof(tip, b)
5:     return (π_b, (π, χ))
6: end function
```

**Algorithm 6** Velvet Infix Verifier

```
1: function VerifyInfixVelvet(b, (π_b, (π, χ)))
2:     tip ← π[−1]
3:     return VerifyInclProof(tip.root_MMR, π_b, b)
4: end function
```

### Maximum delay of an infix proof

The problem with this infix proof construction is that in order to confirm a transaction in a block of interest $b$ the prover has to provide an inclusion proof for that block of interest using the Merkle Tree Root of a more recent honestly generated block included in the infix proof. Obviously, there is a case that there may be a number of consecutive adversarially generated blocks in the chain consisting the prefix of the chain. Thus, no inclusion proof can be provided for any of these blocks until an honestly generated block is added in the chain and, consequently in the infix proof.

We need to estimate an upper bound for the number of rounds a client has to wait until the block of interest is buried under sufficient number of blocks so that he can obtain the infix proof needed with high probability.

Consider $n_b$ is the upper bound on the consecutive rounds when the adversary could possibly append blocks to the chain, without any honestly generated blocks in between. The adversary mines either by following the selfish mining strategy or not. The state of the chain we examine looks like in Figure 17.

Let $p_H$ the probability that a block is generated by an honest party during a round. Then considering that collisions in RO model is a negligible event we have $p_H = (n-t)qp$. Let $p_A$ be the probability that a block is generated by the adversary during a round, then $p_A = tqp$. Consider $N_{bA}$ the random variable for the number of consecutive blocks generated by the adversary in a total of $r$ rounds. Also let $N_{bH}$ the random variable for the number of blocks generated by honest parties in total of $r$ rounds. Then we have that $N_{bA}$, $N_{bH}$ follow the Binomial Distribution with probability $p_A$, $p_H$ respectively.

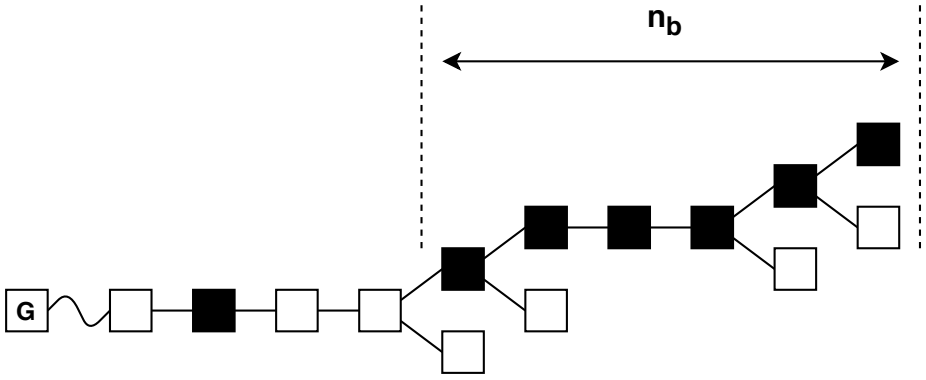


Figure 17: $n_b$ implies the number of consecutive adversarially generated blocks in the chain.Wavy lines imply one or more blocks. Blocks generated by the adversary are colored black.

Because we require consecutive successful rounds for the adversary, we have:

$$Pr[N_{bA} = n_b] = (r - n_b - 1)p_A^{n_b}(1 - p_A)^{r-n_b} \tag{10}$$

while for the honest players we have

$$Pr[N_{bH} \leq n_b] = \sum_{i=0}^{n_b} \binom{n}{k} p_H^i (1 - p_H)^{r-i} \tag{11}$$

If $N_b$ is the random variable denoting the number of consecutive rounds that adversarially generated blocks are appended to the chain, we have:

$$Pr[N_b = n_b] = Pr[N_{bA} = n_b] \cdot Pr[N_{bH} \leq n_b] \tag{12}$$

since the two events are independent.

# List of Figures

# References


[1] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, 2013.

[2] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

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

[4] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. Non-Interactive Proofs of Proof-of-Work. In *International Conference on Financial Cryptography and Data Security*. Springer, 2020.

[5] Ben Laurie, Adam Langley, and Emilia Kasper. Rfc6962: Certificate transparency. *Request for Comments. IETF*, 2013.

[6] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[7] Ling Ren. Analysis of nakamoto consensus, 2019.

[8] Alexei Zamyatin, Nicholas Stifter, Aljosha Judmayer, Philipp Schindler, Edgar Weippl, William Knottenbelt, and Alexei Zamyatin. A wild velvet fork appears! inclusive blockchain protocol changes in practice. In *International Conference on Financial Cryptography and Data Security*. Springer, 2018.

[9] Dionysis Zindros. *Decentralized Blockchain Interoperability*. PhD thesis, Apr 2020.