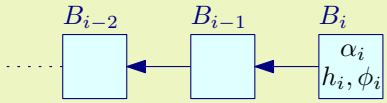
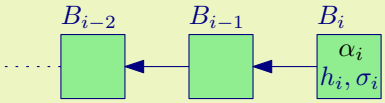


PoW $\phi \Rightarrow$ PoSpace σ



Bitcoin

α_i payload (transactions etc.)

$h_i = \mathsf{H}(B_{i-1})$ hash of previous block

ϕ_i proof of work on challenge $c_i = \mathsf{H}(h_i, \tau_i)$

$\mathsf{PoW.verify}(c_i, \phi_i) = 1 \iff 0.\mathsf{H}(c_i, \phi_i) < 1/D$

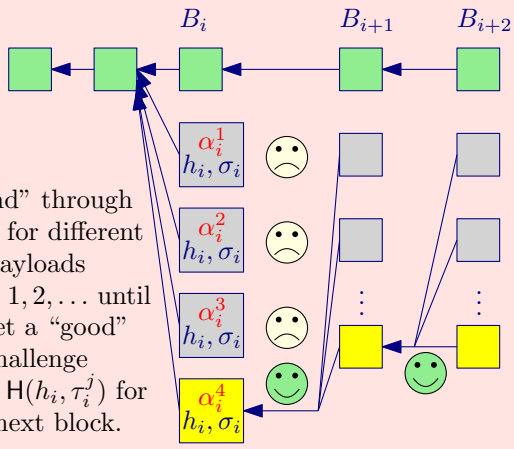
σ_i proof of space on challenge $c_i = \mathsf{H}(h_i, \tau_i)$

$\mathsf{PoSpace.verify}(c_i, \sigma_i) = 1$,

Blocks by honest majority | Adversarial fork | Adversarial blocks that did not make it into the fork

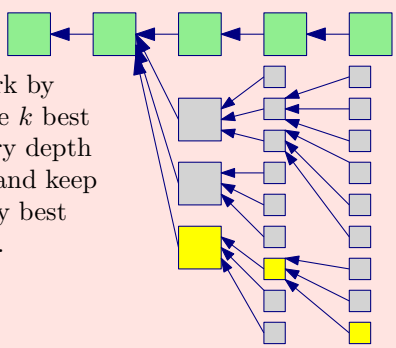
Attack Vector

Grinding



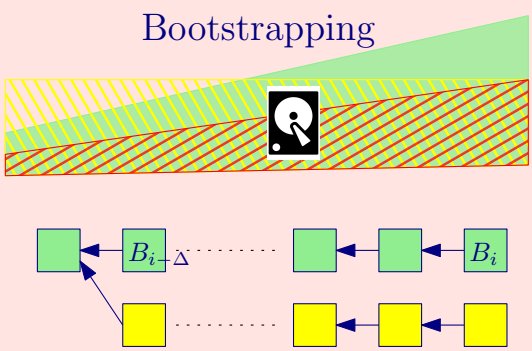
"Grind" through blocks for different payloads $\alpha_i^j, j = 1, 2, \ldots$ until we get a "good" challenge $c_{i+1} = \mathsf{H}(h_i, \tau_i^j)$ for the next block.

Double-Dipping



Create fork by extending the k best heads at every depth (here $k = 9$) and keep the globally best chain.

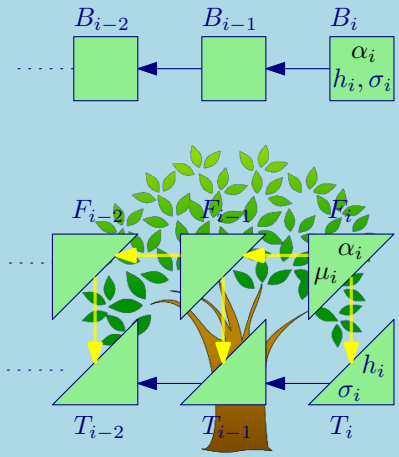
Bootstrapping



Even though the honest parties (green) always control more space than the adversary (red). The adversary can bootstrap a Δ length fork (yellow) that looks as if it was created using the current adversarial space throughout.

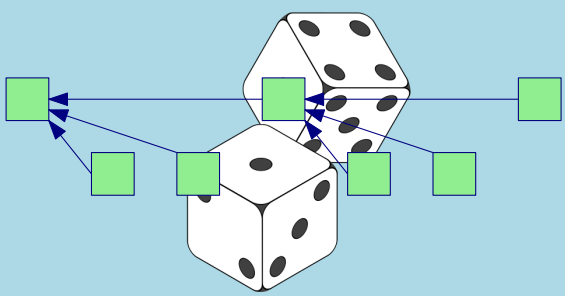
Solution

Splitting the Chain



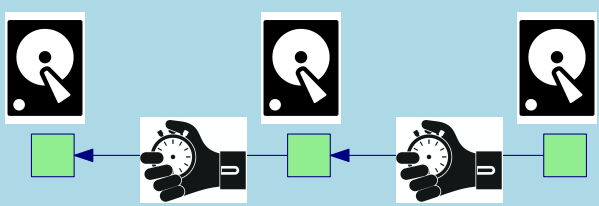
The block $B_i = \{F_i, T_i\}$ is split into trunk and foliage. The "grindable" payload α (transactions, time-stamp) is in the foliage. To bind everything together we add a signature μ_i of the previous foliage block and the current trunk block to the foliage block.

Correlated Randomness



Use the same challenge for $k > 1$ consecutive blocks.

Proofs of Time



Alernate proofs of space with "proofs of time". The cryptographic primitive achieving such proofs is called a verifiable delay function, and basically is an inherently sequential function for which one can provide succinct proofs of correct computation.

The chain can no longer be bootstrapped as computing the VDFs takes time.