

 Mine It – Basic Proof-of-Work Simulation

**Objective/Aim:**  
  
 To simulate Proof-of-Work (PoW) on a local machine by “mining” a toy block: repeatedly hashing data with a

changing nonce until the hash meets a difficulty target (leading zeros).

**Apparatus/Software Used:**

* Laptop
* Word for documentation,
* Proof of work simulator
* Internet for research

**Theory/Concept:**

**What is Proof-of-Work?**

A mechanism where miners prove they expended computation by finding a value (nonce) that makes the

block’s hash satisfy a target condition (e.g., begins with n leading zeros).

**Hash Function (SHA-256):** Deterministic, preimage-resistant; tiny input changes → different outputs.

**Nonce:** Number you tweak on each attempt to change the hash.

**Difficulty:** How hard it is to find a valid hash. In this toy lab, higher difficulty = more required leading

zeros in the hexadecimal hash. Expected trials ≈ 16k16^{k}16k for **k** leading zero **hex** characters.

**Target:** Any hash < threshold; we approximate this by “hash starts with k zeros”.

**Mining Loop:** Increment nonce → hash → check → repeat until success.



**Procedure:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Step 1: Open the Proof-of-Work Simulator in a web browser at:**

**https://blockchain-academy.hs-mittweida.de/2021/05/proof-of-work-simulator/**

**Step 2: In the first block, enter any desired data in the Data field.**

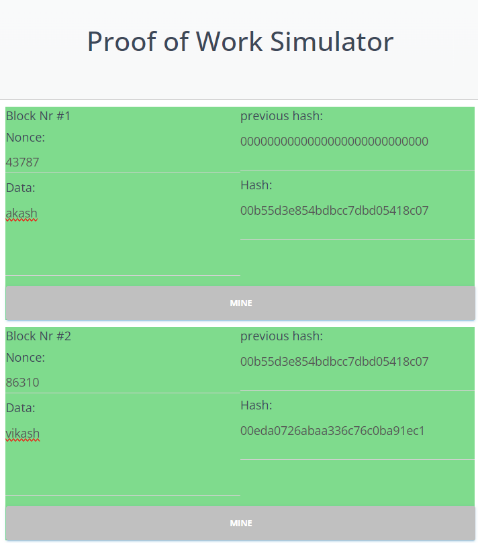
**Step 3:** **Click the Mine button to initiate the PoW process. Observe the nonce, hash, and time taken until the hash meets**

**the difficulty target.**

**Step 4: Repeat the process for each of the subsequent blocks by entering new data into each block’s Data field and mining them one at a time.**

**Step 5: Continue until all five blocks in the simulator have been mined successfully, noting the nonce, hash, and difficulty**

**for each block.**



**Observation:**

**Deterministic Output:** For the same input, SHA-256 always generates the same hash value. Even a

single change—whether a

letter, number, punctuation, or space—produces a completely different hash.

**Avalanche Effect:** A tiny change in the input results in a drastically different output, making patterns

impossible to predict.

**One-Way Function:** The SHA-256 algorithm is irreversible; it is computationally infeasible to retrieve the

original input from the hash.









