Blockchain Foundations
Practice Final
2026

1. The exam has 5 questions with a total of 124 points. You have 3 hours to take the exam. Questions have different numbers of points so please allocate your time to each question accordingly.
2. Please write the answer to each question on a separate page and upload a photo or scan to Gradescope.
3. **All answers should be justified, unless otherwise stated.**
4. The exam is open-book, open-notes, open-internet.

Good luck!

(60 points) Problem 1

For the following questions, choose the one most fitting answer among the four choices. No justifications are required for this problem. 2 points for a correct answer, 0 points for an incorrect answer. 0.5 point for leaving the answer blank. Knowing you don't know something has value.

1. You are a passive observer and you notice that the ledger has not been including any new transactions over the past week.

   (a) You can deduce that safety is lost.

   (b) You can deduce that liveness is lost.

   (c) You can deduce that both safety and liveness are lost.

   (d) You can make no such deductions.

2. Consider a proof-of-work longest chain protocol with $t = n/3$. What is the minimum chain quality an adversary can cause over the long run?

   (a) $\mu = 0$

   (b) $\mu = 1/4$

   (c) $\mu = 1/3$

   (d) $\mu = 1/2$

3. Consider a Merkle tree with 1024 leaves that uses SHA256. How long is the proof size for one inclusion in this tree, in bits?

   (a) 1024

   (b) 10

   (c) 2560

   (d) 262,144

4. Which of the following assurances does an existentially unforgeable signature scheme give?

   (a) Given an existing message and a correct signature on it, the adversary cannot create a new, different signature on the same message verifiable under the same public key.

   (b) Given a message and a correct signature, the adversary cannot recover the secret key.

   (c) Given just a correct signature and a public key, the adversary cannot recover the message.

   (d) All of the above.

5. Consider a $\kappa = 256$ proof-of-work puzzle with $T = 2^{192}$. How many queries do you expect to need before you get a solution?

   (a) 256

   (b) 64

   (c) 2^{192}

   (d) 2^{64}

6. Consider a $\kappa = 256$ proof-of-stake longest chain puzzle with $T_p = 2^{236}$. There are $n = 1{,}000{,}000$ nodes. What is the expected number of nodes that will win in a given round?

   (a) 0

   (b) 1

   (c) 10

   (d) unbounded

7. Consider a $\kappa = 256$ proof-of-stake longest chain puzzle with $T_p = 2^{236}$. There are $n = 1{,}000{,}000$ nodes. What is the expected number of valid blocks that can be proposed in a given round?

   (a) 0

   (b) 1

   (c) 10

   (d) unbounded

8. In order to ensure ledger safety in an honest majority setting, the macroeconomic policy of a cryptocurrency must mandate that:

   (a) Total coin supply rate must be non-increasing over time.

   (b) Total coin supply rate must be strictly decreasing over time.

   (c) Total coin supply rate must be non-decreasing over time.

   (d) None of the above.

9. In a proof-of-stake longest chain protocol, which of the following adversarial bounds are sufficient to achieve ledger safety and liveness respectively?

   (a) $t < n/3$ for both safety and liveness.

   (b) $t < 2n/3$ for both safety and liveness.

   (c) $t < n/3$ for safety, but $t < 2n/3$ for liveness.

   (d) $t > n/3$ for safety, but $t > 2n/3$ for liveness.

10. If H is a random oracle, then $G(x) = H(H(x))$ is:

(a) Collision resistant, but not preimage resistant.
(b) Preimage resistant, but not second preimage resistant.
(c) Behaving like a random oracle.
(d) None of the above.

11. Under a block size limit (in bytes), a rational miner prioritizes transactions by:

    (a) Their size, in bytes, in increasing order.
    (b) Their fees, in decreasing order.
    (c) The ratio fees / byte, in increasing order.
    (d) None of the above.

12. Let G be a second-preimage resistant hash function. Consider the hash function $H(x)$ that prepends the fixed string "0110" to the output of $G(x)$. The function H might *fail* to be:

    (a) Collision resistant
    (b) Preimage resistant
    (c) Second-preimage resistant
    (d) None of the above

13. The data that needs to be downloaded by an SPV light wallet holding an address that has been used to send or receive money a constant number of times and is synchronizing for the first time is:

    (a) Linear in the chain length, but only logarithmic in the number of transactions per block.
    (b) Linear in the chain length and linear in the number of transactions per block.
    (c) Logarithmic in the chain length, but linear in the number of transactions per block.
    (d) Logarithmic in the chain length and logarithmic in the number of transactions per block.

14. When the hashrate of the network increases, the variable difficulty proof-of-work protocol:

    (a) Decreases T so that participants are incentivized to decrease q.
    (b) Decreases T so that f is decreased over the long run and convergence opportunities become denser and denser.
    (c) Decreases p in order to keep f constant over the long run.
    (d) Increases T in order to raise the difficulty.

15. Which ledger virtues can a temporary majority adversarial miner break in a way that is unhealable (they are not regained after any point in time) in the proof-of-work protocol?

    (a) Safety.
    (b) Liveness.
    (c) Both.
    (d) Neither.

16. Our Marabu protocol was attacked by a charming selfish mining adversary building a new chain from genesis. Many students are falsely claiming they are this Adversary. What is an appropriate way for the adversary to prove her identity?

    (a) Open source her mining code on GitHub.
    (b) Place her SUID in the "note" field of a new block on top of the adversarial chain.
    (c) Sign her SUID using the coinbase key of the first block in the attack.
    (d) Generate a new public/private key pair and use it to sign a message containing both the latest adversarial tip as well as her SUID concatenated together.

17. Which of the following functions is negligible?

    (a) $1/n^2$
    (b) $e^{-n/3}$
    (c) $1/\log(\log(n))$
    (d) All of the above.

18. Double spending transactions, in which some transaction believed to be confirmed is later reverted and becomes unconfirmed, are avoided:

    (a) In both the UTXO and the accounts model by a signature.
    (b) In both the UTXO and the accounts model by a nonce.
    (c) In the UTXO model by a signature, and in the accounts model by the nonce.
    (d) By the underlying blockchain.

19. Why can't a coinbase transaction generate more money than the macroeconomic policy mandates?

    (a) It will invalidate the coinbase signature, and this will be checked by every node before propagating the block further.
    (b) It will invalidate the Law of Conservation, that input values must exceed output values.
    (c) Every node will do a hard-coded check that the output value is as required.
    (d) The coinbase transaction will be spending money that is not in the UTXO set.

20. During a chain reorg, transactions evicted from the chain are:

    (a) Placed back into the mempool if still valid.
    (b) Evicted from the chain, but manually inserted into the ledger.
    (c) Placed in the next block template, but not in the mempool.
    (d) Discarded.

21. Imagine a network where suddenly the non-eclipsing assumption no longer holds. What is the worst thing a non-mining adversary can do to a proof-of-work longest chain protocol?

    (a) Transactions get confirmed, and they are never reverted in the future.
    (b) Transactions get confirmed, but may be reverted in the future.
    (c) Transactions are included in the chain, but may not be confirmed.
    (d) No transactions ever make it to the chain.

22. How are peers discovered in a peer-to-peer network such as a blockchain network?

    (a) The client connects to a secure HTTPS server which gives us a list of peers.
    (b) The peers are discovered by connecting to a decentralized database such as a shared instance of MySQL, Postgres, or Mongodb that contains a list of all known peers. The database is replicated to ensure reliability.
    (c) The list of peers is fixed and includes at least one known trustworthy honest peer such as the IP of one of the developers. This list is never updated to avoid introducing adversarial peers.
    (d) Some peers are hardcoded in the code so that we can connect to them initially. The rest are discovered by asking our peers for their peers.

23. What is the best way to avoid denial-of-service attacks of fake blocks?

    (a) Check the proof-of-work first, then download the transactions, then download the parent.
    (b) Download and validate the transactions first, then check the proof-of-work, then download the parent.
    (c) Download and validate the transactions first, then download the parent, and finally check the proof-of-work.
    (d) Make sure the parent is available first by recursively downloading it and validating it, then check the proof-of-work, and only afterwards download the transactions.

24. A malicious full node, when asked by an SPV node, pretends a transaction is in the chain, while it is not. What will the SPV node do?

    (a) Accept the transaction, as it cannot check it itself; it relies on the full node for security.

(b) Ask other full nodes if they have accepted the transaction and take a majority vote.

(c) Reject the transaction, as the Merkle proof does not check out.

(d) Reject the transaction, because the transaction signature does not validate.

25. Why does the Chain Growth property require a minimum number of rounds parameter s?

    (a) So that the adversary has enough time to run.

    (b) So that the expectation $\mathbb{E}[X]$ attains the lower bound we require.

    (c) So that the Chernoff bound can be applied to the number of successful rounds X.

    (d) So that the adversarially successful queries Z concentrate to a value lower than the convergence opportunities Y.

26. When representing the UTXO model as a State Machine Replication problem, what is the type of the output of the transition function $\delta(st, tx)$?

    (a) A UTXO transaction.

    (b) The current balances of the whole system, together with the nonce of each account.

    (c) A set of unspent outputs.

    (d) True or false, depending on whether the transaction can be applied to the previous state.

27. In a proof-of-work longest chain protocol, who can predict which miner will win the next block?

    (a) Anybody.

    (b) The adversary.

    (c) The miner who will win the next block.

    (d) Nobody.

28. In a proof-of-stake longest chain protocol using a hash function for the puzzle, who can predict which node will win in the next round?

    (a) Anybody who knows the public keys of the nodes.

    (b) The adversary.

    (c) The node who will win in the next round.

    (d) Nobody.

(8 points) Problem 2

Consider a proof-of-work longest chain protocol with a 1/3 adversary in a population of $n = 3$ nodes with a hash rate of $q = 3$. The security parameter is $\kappa = 256$.

1. (2 points) What is the honest advantage δ?
2. (2 points) Choose numeric parameters ϵ and f to ensure safety and liveness.
3. (2 points) Calculate the exact numeric probability p of a successful query.
4. (2 points) Calculate a numeric value for the mining target T to match the above.

You can use a calculator such as Python and round your numbers to three significant digits.

(20 points) Problem 3

Consider the longest chain proof-of-stake protocol we studied in class. We will look at an instantiation where $\Delta = 1$ second, $n = 100$, $\kappa = 128$, $T_p = 2^{118}$. For answering the questions below, you are free to choose the parameter k in the confirmation rule. You can use a calculator such as Python and round your numbers to three significant digits.

1. First suppose all 100 nodes are honest.
   (a) (1 point) Compute the expected growth rate of the longest chain, in blocks per second.
   (b) (1 point) Compute the expected growth rate of the k-confirmed chain, in blocks per second. (The k-confirmed chain is the portion $C[:-k]$.)
   (c) (2 points) If we double T_p, does the expected growth rate of the longest chain double, more than double, or less than double? What about the growth rate of the k-confirmed chain?
2. Now suppose 20 of the 100 nodes are adversary, the other 80 are honest.
   (a) (1 point) Compute a tight lower bound on the expected growth rate of the longest chain. (A "lower bound" means that it is a lower bound irrespective of the adversary's attack strategy; "tight" means that the lower bound is attainable for some adversary's attack strategy.)
   (b) (1 point) Compute a tight lower bound on the expected growth rate of the k-confirmed chain.
   (c) (2 points) Is the protocol safe? Is the protocol live?
3. Now suppose 20 nodes are still adversary, but instead of having the full 80 honest nodes online, 30 of them decide not to participate in the protocol and went on vacation to the Bermudas. However, the protocol designer does not know this and the protocol parameters are not adjusted.
   (a) (1 point) Compute a tight lower bound on the expected growth rate of the longest chain.
   (b) (1 point) Compute a tight lower bound on the growth rate of the k-confirmed chain.
   (c) (2 points) Is the protocol safe? Is the protocol live?
4. Now suppose a further 35 honest nodes went on vacation.
   (a) (1 point) Compute a tight lower bound on the expected growth rate of the longest chain.
   (b) (1 point) Compute a tight lower bound on the expected growth rate of the k-confirmed chain.
   (c) (2 points) Is the protocol safe? Is the protocol live?

5. (4 points) A protocol is said to be *available* if it is safe and live whenever the number of honest nodes online is greater than the number of adversary nodes. Is the longest chain PoS protocol available?

(18 points) Problem 4

Consider the Streamlet protocol we studied in class.

1. (2 points) Streamlet is said to be *partition tolerant* whenever the number of honest nodes exceeds $2n/3$, where n is the total number of nodes. Explain what that means.

2. (2 points) Streamlet is said to be 1/3-*accountable*. Explain what that means.

3. Now suppose $\Delta = 1$ second and $n = 100$ nodes and the nodes are all honest.

   (a) (1 point) Compute the expected growth rate of the longest notarized chain (in blocks per second).

   (b) (1 point) Compute the expected growth rate of the finalized chain (in blocks per second).

4. Now suppose 20 of the 100 nodes are adversary and the other 80 are honest.

   (a) (1 point) Compute a tight lower bound on the expected growth rate of the longest notarized chain. (A "lower bound" means that it is a lower bound irrespective of the adversary's attack strategy; "tight" means that the lower bound is attainable for some adversary's attack strategy.)

   (b) (1 point) Compute a lower bound on the expected growth rate of the finalized chain. (Your lower bound does not need to be tight. However if the lower bound is not tight, it must be positive.)

   (c) (2 points) Is the protocol safe? Is it live?

5. Now suppose 20 nodes are still adversary, but instead of having the full 80 honest nodes online, 30 of them decide not to participate in the protocol and went on vacation to the Bermudas. However, the protocol designer does not know this and the protocol parameters are not adjusted.

   (a) (1 point) Compute a tight lower bound on the expected growth rate of the longest notarized chain.

   (b) (1 point) Compute a lower bound on the expected growth rate of the finalized chain. (Your lower bound does not need to be tight. However if the lower bound is not tight, it must be positive.)

   (c) (2 points) Is the protocol safe? Is the protocol live?

6. (4 points) A BFT protocol is said to be *available* if it is safe *and* live whenever the number of honest nodes online is greater than twice the number of adversary nodes. Is Streamlet available?

(18 points) Problem 5

Answer the following questions in the Backbone model with static difficulty.

1. Consider executions with parameters $(n, q, t, f, \epsilon, T, \mu, \ell, s, \tau, u, k)$ of your choice, *without* honest majority, but with $n - t > 0$.

   (a) (4 points) Describe an execution where Safety is violated.

   (b) (4 points) Describe an execution where Liveness is violated.

   (c) (1 point) Is it possible to have an execution with a Safety violation if chain quality is *not* violated?

   (d) (1 point) Is it possible to have an execution with a Liveness violation if chain quality is *not* violated?

   For questions (a) and (b) above: What is the strategy of the adversary? What blocks and what transactions must the adversary produce to cause this ledger virtue violation? Draw the block tree and timeline of the execution illustrating which round each block was mined in, which transactions are included in which block, whether a block was computed by an honest or an adversarial party, and which honest parties have adopted which chain. You do not need to calculate values of parameters that are not necessary to support the respective statement.

2. Consider an execution with $n = 3$ parties of which $t = 1$ is adversarial, target $T = 2^{226}$, security parameter $\kappa = 256$ and hash rate $q = 1$, and $k = 6$.

   (a) (2 points) Calculate the numeric probability of a successful round.

   (b) (2 points) Calculate the numeric probability of a convergence opportunity.

   (c) (2 points) What is the numeric probability that the first 10 rounds are all successful?

   You can use a calculator such as Python and round your numbers to three significant digits.

3. (2 points) In the above scenario, we give the adversary the fictitious ability to "snoop" all the queries to the Random Oracle and to choose whatever κ-bit answer she wishes to one (honest or adversarial) fresh query *per round*. The Random Oracle continues to use its cache to respond consistently. Can this adversary break common prefix? What strategy should she follow?

4. (3 bonus points) In the above scenario, we give the adversary the fictitious ability to "snoop" all the queries to the Random Oracle and to choose whatever κ-bit answer she wishes to one (honest or adversarial) fresh query *per execution*. The Random Oracle continues to use its cache to respond consistently. Can this adversary break common prefix? What strategy should she follow?

Reference

Variables

- κ: The security parameter
- $\mathcal{A}$: The adversary
- Π: The honest protocol
- $\mathcal{G}$: The genesis block
- Δ: The network delay (in backbone, $\Delta = 1$)
- H: The hash function
- n: The total number of parties
- t: The adversarial number of parties
- q: Hash rate of one party per round
- T: The mining target
- p: Probability of a successful query
- δ: The honest advantage
- k: Common prefix parameter
- μ: Chain quality parameter (the honest ratio of blocks)
- ℓ: Chain quality chunk length (in blocks)
- τ: Chain growth rate (in blocks per round)
- s: Chain growth duration (in rounds)
- f: Probability of successful round
- ϵ: Chernoff bound error
- λ: Chernoff bound duration
- X: Successful round indicator
- Y: Convergence opportunity indicator
- Z: Adversarially successful query indicator

Formulae

- The honest majority assumption: $t < (1 - \delta)(n - t)$.
- The balancing equation: $3f + 3\epsilon \leq \delta$.
- The proof-of-work equation: $H(B) \leq T$.
- The proof-of-stake equation: $H(s_0 \,\|\, pk \,\|\, r) \leq T_p$.

Algorithms

Algorithm 1 The collision resistance game.

```
1: function $\textsc{Collision}_{H,\mathcal{A}}(\kappa)$
2:     $x_1, x_2 \leftarrow \mathcal{A}(1^\kappa)$
3:     return $x_1 \neq x_2 \wedge H_\kappa(x_1) = H_\kappa(x_2)$
4: end function
```

Algorithm 2 The preimage resistance game.

```
1: function $\textsc{Preimage}_{H,\mathcal{A}}(\kappa)$
2:     $x \overset{\$}{\leftarrow} \{0,1\}^{2\kappa}$
3:     $y \leftarrow H_\kappa(x)$
4:     $x^* \leftarrow \mathcal{A}(y)$
5:     return $H_\kappa(x^*) = H_\kappa(x)$
6: end function
```

Algorithm 3 The second preimage resistance game.

```
1: function $\textsc{2nd-Preimage}_{H,\mathcal{A}}(\kappa)$
2:     $x \overset{\$}{\leftarrow} \{0,1\}^{2\kappa}$
3:     $x' \leftarrow \mathcal{A}(x)$
4:     return $H_\kappa(x) = H_\kappa(x') \wedge x \neq x'$
5: end function
```

Algorithm 4 The existential forgery game for a signature scheme (Gen, Sig, Ver).

```
function existential-forgery-game$_{Gen,Sig,Ver,\mathcal{A}}(\kappa)$
    $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
    $M \leftarrow \emptyset$
    function $\mathcal{O}$(m)
        $M \leftarrow M \cup \{m\}$
        return $\mathsf{Sig}(sk, m)$
    end function
    $m, \sigma \leftarrow \mathcal{A}^{\mathcal{O}}(pk)$
    return $\mathsf{Ver}(pk, \sigma, m) \wedge m \notin M$
end function
```

Algorithm 5 The Random Oracle

```
$r \leftarrow 0$
$\mathcal{T} \leftarrow \{\}$                              ▷ Initiate Cache
$Q \leftarrow 0$                                     ▷ $q$ for honest parties, $qt$ for adversary
function $H_\kappa(x)$
    if $x \notin \mathcal{T}$ then                         ▷ First time being queried
        if $Q = 0$ then                        ▷ Out of Queries
            return $\perp$
        end if
        $Q \leftarrow Q - 1$
        $\mathcal{T}[x] \xleftarrow{\$} \{0,1\}^\kappa$
    end if
    return $\mathcal{T}[x]$                          ▷ Return value from Cache
end function
```

Algorithm 6 The environment.

```
$r \leftarrow 0$
function $\mathcal{Z}^{n,t}_{\Pi,\mathcal{A}}(1^\kappa)$
    $\mathcal{G} \xleftarrow{\$} \{0,1\}^\kappa$                                        ▷ Genesis block
    for $i \leftarrow 1$ to $n - t$ do                              ▷ Boot stateful honest parties
        $P_i \leftarrow$ new $\Pi(\mathcal{G})$
    end for
    $A \leftarrow$new $\mathcal{A}(\mathcal{G}, n, t)$                                  ▷ Boot stateful adversary
    $\overline{M} \leftarrow []$                                              ▷ 2D array of messages
    for $i \leftarrow 1$ to $n - t$ do
        $\overline{M}[i] \leftarrow []$                      ▷ Each honest party has an array of messages
    end for
    while $r < \mathsf{poly}(\kappa)$ do                                      ▷ Number of rounds
        $r \leftarrow r + 1$
        $M \leftarrow \emptyset$
        for $i \leftarrow 1$ to $n - t$ do                        ▷ Execute honest party $i$ for round $r$
            $Q \leftarrow q$    ▷ Maximum number of oracle queries per honest party (Section 2)
            $M \leftarrow M \cup \{P_i.\mathsf{execute}^H(\overline{M}[i])\}$          ▷ Adversary collects all messages
        end for
        $Q \leftarrow tq$                              ▷ Max number of Adversarial oracle queries
        $\overline{M} \leftarrow A.\mathsf{execute}^H(M)$                     ▷ Execute rushing adversary for round $r$
        for $m \in M$ do                          ▷ Ensure all parties will receive message $m$
            for $i \leftarrow 1$ to $n - t$ do
                $\mathsf{assert}(m \in \overline{M}[i])$                          ▷ Non-eclipsing assumption
            end for
        end for
    end while
end function
```

Algorithm 7 The honest party

```
$\mathcal{G} \leftarrow \epsilon$
function Constructor($\mathcal{G}'$)
    $\mathcal{G} \leftarrow \mathcal{G}'$                                  ▷ Select Genesis Block
    $\mathcal{C} \leftarrow [\mathcal{G}]$                                ▷ Add Genesis Block to start of chain
    round $\leftarrow 1$
end function
function Execute($1^\kappa$)
    $\tilde{\mathcal{C}} \leftarrow \text{maxvalid}(\mathcal{C}, \bar{M}[i])$            ▷ Adopt Longest Chain in the network
    if $\tilde{\mathcal{C}} \neq \mathcal{C}$ then
        $\mathcal{C} \leftarrow \tilde{\mathcal{C}}$
        Broadcast($\mathcal{C}$)                              ▷ Gossip Protocol
    end if
    $x \leftarrow$ Input()                                  ▷ Take all transactions in mempool
    $B \leftarrow \text{PoW}(x, H(\mathcal{C}[-1]))$
    if $B \neq \bot$ then                                  ▷ Successful Mining
        $\mathcal{C} \leftarrow \mathcal{C}||B$                               ▷ Add block to current longest chain
        Broadcast($\mathcal{C}$)                              ▷ Gossip protocol
    end if
    round $\leftarrow$ round$+1$
end function
function Read
    $x \leftarrow \epsilon$                                       ▷ Instantiate transactions
    for $B \in \mathcal{C}$ do
        $x \leftarrow x||B.x$                    ▷ Extract all transactions from each block in the chain
    end for
    return $x$
end function
```

Algorithm 8 Mining

```
1: function $\textsc{pow}_{H,T,q}(x, s)$
2:     $ctr \xleftarrow{\$} \{0,1\}^\kappa$                              ▷ Randomly sample Nonce
3:     for $i \leftarrow 1$ to $q$ do                         ▷ Number of available queries per party
4:         $B \leftarrow s||x||ctr$                                    ▷ Create block
5:         if $H(B) \leq T$ then                                ▷ Successful Mining
6:             return $B$
7:         end if
8:         ctr ← ctr +1
9:     end for
10:    return $\perp$                                         ▷ Unsuccessful Mining
11: end function
```

Algorithm 9 The longest chain rule

```
1: function $\textsc{maxvalid}_{\mathcal{G},\delta(\cdot)}(\overline{C})$
2:     $C_{\mathsf{max}} \leftarrow [\mathcal{G}]$                         ▷ Start with current adopted chain
3:     for $C \in \overline{C}$ do          ▷ Iterate for every chain received through gossip network
4:         if $\mathsf{validate}_{\mathcal{G},\delta(\cdot)}(C) \wedge |C| > |C_{\mathsf{max}}|$ then          ▷ Longest Chain Rule
5:             $C_{\mathsf{max}} \leftarrow C$
6:         end if
7:     end for
8:     return $C_{\mathsf{max}}$
9: end function
```

Algorithm 10 Chain Validation

```
function VALIDATE$_{\mathcal{G},\delta(\cdot)}(C)$
    if $C[0] \neq \mathcal{G}$ then                          ▷ Check that first block is Genesis
        return false
    end if
    st ← st$_0$                                              ▷ Start at Genesis state
    $h \leftarrow H(C[0])$
    st ← $\delta^*(st, C[0].x)$
    for $B \in C[1{:}]$ do                                   ▷ Iterate for every block in the chain
        $(s, x, ctr) \leftarrow B$
        if $H(B) > T \vee s \neq h$ then                     ▷ PoW check and Ancestry check
            return false
        end if
        st ← $\delta^*(st, B.x)$        ▷ Application Layer: update UTXO & validate transactions
        if $st = \perp$ then
            return false                                     ▷ Invalid state transition
        end if
        $h \leftarrow H(B)$
    end for
    return true
end function
```

Chain Virtues

1. **Common Prefix (k).** $\forall$ honest parties P_1, P_2 adopting chains C_1, C_2 at any rounds $r_1 \leq r_2$ respectively, $C_1[:-k] \preceq C_2$ holds.

2. **Chain Quality (μ, ℓ).** $\forall$ honest party P with adopted chain C, $\forall i$ any chunk $C[i : i+\ell]$ of length $\ell > 0$ has a ratio of honest blocks μ.

3. **Chain Growth (τ, s).** $\forall$ honest parties P and $\forall r_1, r_2$ with adopted chain C_1 at round r_1 and adopted chain C_2 at round $r_2 \geq r_1 + s$, it holds that $|C_2| \geq |C_1| + \tau s$.

Ledger Virtues

- **Safety:** For all honest parties P_1, P_2, and rounds r_1, r_2, $L_{r_1}^{P_1}$ is a prefix of $L_{r_2}^{P_2}$ or vice versa.

- **Liveness(u)**: If all honest parties attempt to inject a transaction tx at rounds $r, ..., r+u$, then for all honest parties P, tx will appear in L_{r+u}^{P}.