Blockchain Foundations **NTUA, Spring 2026**

Theory Workshop 3

Problem 1

In class, we studied Merkle trees, which let us prove inclusion of a single leaf with a logarithmic-sized proof. In many applications, however, we are not interested in a single leaf but rather in an *aggregate statement* over a contiguous range of leaves. For instance, given a block containing a vector of n transactions $\bar{x} = (x_0, x_1, \ldots, x_{n-1})$ with associated fees $f_0, f_1, \ldots, f_{n-1}$, a light client might want a succinct proof that the total fees collected from transactions $\bar{x}[a{:}b]$ equals some claimed value s, i.e. that $\sum_{i=a}^{b-1} f_i = s$. A naive solution would be to provide a separate Merkle inclusion proof for each transaction in the range and let the verifier sum the fees themselves; this requires $\Theta((b-a)\log n)$ hashes in the worst case, which is linear in the size of the range.

A better solution is to use a *Merkle Segment Tree*. A segment tree is a complete binary tree built on top of an array (here, the array of transactions) in which every internal node represents the contiguous range of leaves in its subtree. In a classical (non-cryptographic) segment tree, each node stores a value obtained by applying an associative *aggregation function* (the sum, the maximum, the minimum, etc.) to the values of its two children; this allows answering range queries over any interval $[a{:}b]$ in $O(\log n)$ time by decomposing the interval into $O(\log n)$ *canonical* subtree ranges whose union is exactly $[a{:}b]$.

A Merkle Segment Tree combines this idea with the authentication mechanism of a Merkle tree: each node stores *both* the aggregate of its subtree *and* a hash that cryptographically commits to the entire subtree (its structure, its leaves, and its aggregates). Concretely, for the sum aggregation:

- Each leaf i stores the pair (s_i, v_i) where $s_i = f_i$ is the aggregate (here, just the fee itself) and $v_i = H(f_i \| H(x_i))$ is the leaf's hash commitment.
- Each internal node u with left child l and right child r stores the pair (s_u, v_u) where $s_u = s_l + s_r$ and $v_u = H(s_l \| v_l \| s_r \| v_r)$.

The root value v_{root}, together with the root aggregate s_{root}, are published as the commitment to the whole block (analogous to the Merkle root).

Describe a proof-of-inclusion scheme that allows a prover to convince a verifier (who only knows v_{root} and s_{root}) that the sum of fees of transactions $\bar{x}[a{:}b]$ equals a claimed value s. Specify exactly:

1. what data the prover sends to the verifier,
2. the verification algorithm, and
3. the asymptotic size of the proof as a function of n and $b - a$.

A classical segment tree answers "sum of fees in $[a : b)$" by carving the range into $O(\log n)$ precomputed subtree-sums and adding them up. A Merkle segment tree does the same thing,

but each node additionally carries a hash that commits to its entire subtree, so a light client can be *convinced* of the sum without trusting the prover. The prover sends the $O(\log n)$ subtree-sums whose ranges partition $[a : b)$, together with just enough sibling hashes to let the verifier recompute the root commitment. The verifier checks (i) that the supplied sums hash up to the published root, and (ii) that they add up to the claimed value s.

Setup and notation. Assume n is a power of two, so that the segment tree is a complete binary tree of height $\log_2 n$ with the transactions $x_0, \ldots, x_{n-1}$ at the leaves (the construction extends to arbitrary n by padding with neutral leaves $(0, H(\bot))$). Every node u corresponds to a contiguous range $[\ell_u : r_u)$ of leaves; a leaf at position i has range $[i : i+1)$ and the root has range $[0 : n)$. Each node stores the pair (s_u, v_u) defined in the problem statement. A node u is *contained* in $[a : b)$ when $[\ell_u : r_u) \subseteq [a : b)$, and *disjoint* when $[\ell_u : r_u) \cap [a : b) = \emptyset$.

Canonical decomposition. Starting at the root, recurse into a node u only if $[\ell_u : r_u)$ partially overlaps $[a : b)$; stop as soon as the node is fully contained (output it) or fully disjoint (discard it). The resulting set

$$\mathcal{C} = \{u_1, u_2, \ldots, u_m\}, \qquad \bigsqcup_{u \in \mathcal{C}} [\ell_u : r_u) = [a : b),$$

is the unique minimal canonical decomposition: no proper ancestor of any u_k is itself fully contained in $[a : b)$. A standard counting argument shows that at each of the $\log_2 n$ levels at most two nodes partially overlap $[a : b)$, so $m \leq 2 \log_2 n$.
The crucial observation is that this decomposition is a deterministic function of (a, b, n) alone: no fee data, no hashes, just index arithmetic on ranges. The verifier can therefore reproduce the positions of the canonical nodes on its own, and the prover only needs to send the *values* stored at those positions, in a fixed (left-to-right) order.

Proof contents. The prover sends two things, together with the public inputs (a, b, s, n):

1. the canonical pairs $(s_{u_1}, v_{u_1}), \ldots, (s_{u_m}, v_{u_m})$, in left-to-right order of their ranges;
2. a multi-authentication path π that lets the verifier recompute v_{root} by folding the canonical pairs upward.

To define π, let $\mathcal{P}$ be the set of *proper* ancestors of the canonical nodes (the union of the root-paths from each u_k to the root, excluding the u_k themselves). For every node $w \in \mathcal{P}$ whose sibling $\tilde{w}$ does *not* lie in $\mathcal{C} \cup \mathcal{P}$, the prover includes the sibling's pair $(s_{\tilde{w}}, v_{\tilde{w}})$. Sibling pairs of nodes that already lie in $\mathcal{C} \cup \mathcal{P}$ are omitted, because the verifier will reconstruct them during folding.

Verification algorithm. Given (a, b, s), the published commitment $(s_{\text{root}}, v_{\text{root}})$, and the proof, the verifier proceeds in three phases.

Algorithm 1 Range-sum verification

Phase 1 — Reconstruct canonical positions.
Run the recursive decomposition on (a, b, n) to obtain $\mathcal{C}^\star = (u_1^\star, \ldots, u_{m^\star}^\star)$ in left-to-right order.
Reject if the prover supplied $m \neq m^\star$ pairs.
Assign the k-th supplied pair (s_{u_k}, v_{u_k}) to $u_k^\star$.
Phase 2 — Fold to the root.
Initialize $W \leftarrow \{(u_k, s_{u_k}, v_{u_k})\}_{k=1}^{m}$.
while $|W| > 1$ **do**
    Pick the entry $(u, s_u, v_u) \in W$ with the largest depth.
    Let $\tilde{u}$ be the sibling of u.
    if $(\tilde{u}, s_{\tilde{u}}, v_{\tilde{u}}) \in W$ **then**
        use that triple
    else if $(s_{\tilde{u}}, v_{\tilde{u}}) \in \pi$ **then**
        use the pair from π
    else reject
    end if
    Determine left/right from tree positions; set $s_{\text{par}} \leftarrow s_{\text{left}} + s_{\text{right}}$ and $v_{\text{par}} \leftarrow H(s_{\text{left}} \,\|\, v_{\text{left}} \,\|\, s_{\text{right}} \,\|\, v_{\text{right}})$.
    Remove $(u, \cdot, \cdot)$ and $(\tilde{u}, \cdot, \cdot)$ from W; insert $(\text{parent}(u), s_{\text{par}}, v_{\text{par}})$.
end while
Let $(\text{root}, s^\star, v^\star)$ be the unique entry of W.
Reject if $v^\star \neq v_{\text{root}}$ or $s^\star \neq s_{\text{root}}$.
Phase 3 — Sum check.
Reject if $\sum_{k=1}^{m} s_{u_k} \neq s$.
Accept.

Why the sibling of u may already be in W. When two canonical nodes happen to be siblings, the deepest-first rule pulls them out together and combines them directly into their parent, with no entry needed from π. More generally, after several folding steps, two nodes originating from different parts of $\mathcal{C}$ may be promoted to sibling positions at some higher level; again π is not consulted. These "free" combinations are precisely why $|\pi|$ stays small.

Correctness. For an honestly generated proof, the supplied (s_{u_k}, v_{u_k}) are the true stored values and π contains exactly the siblings the verifier needs. Phase 2 therefore replays the bottom-up construction of the tree along the union of root-paths from $\mathcal{C}$, so $v^\star = v_{\text{root}}$ and $s^\star = s_{\text{root}}$. By definition of the canonical decomposition, the leaves under $\mathcal{C}$ are exactly $\bar{x}[a : b)$, hence $\sum_k s_{u_k}$ equals the true fee sum, which equals s.

Soundness. Assume H is collision-resistant. Suppose a cheating prover makes the verifier accept on a wrong claim $s' \neq \sum_{i=a}^{b-1} f_i$. By Phase 1 the supplied pairs are anchored at the true canonical positions. By Phase 3 $\sum_k s_{u_k} = s' \neq \sum_{i=a}^{b-1} f_i$, so at least one canonical node

$u_{k^\star}$ has a supplied aggregate $\hat{s}_{u_{k^\star}} \neq s_{u_{k^\star}}$, where $s_{u_{k^\star}}$ is the honest committed value. The supplied pair at $u_{k^\star}$ therefore differs from the honest pair in at least the first component. Walking up from $u_{k^\star}$ to the root, every parent commitment is recomputed as $v_{\text{par}} = H(s_{\text{left}} \,\|\, v_{\text{left}} \,\|\, s_{\text{right}} \,\|\, v_{\text{right}})$. Phase 2 ends with $v^\star = v_{\text{root}}$, which equals the honest root commitment. So there is a *lowest* ancestor w on this path at which the recomputed pair $(\hat{s}_w, \hat{v}_w)$ agrees with the honest (s_w, v_w), while at the child just below w the pairs disagree. The hash inputs at w — the prover's versus the honest ones — therefore differ in at least one of the four fields, yet they produce the same v_w. This is a collision for H, contradicting collision-resistance. □

Proof size. The proof carries:

- $m \leq 2\log_2 n$ canonical pairs (s_{u_k}, v_{u_k});
- the auxiliary siblings in π. To bound $|\pi|$, observe that the ancestors $\mathcal{P}$ of canonical nodes lie on at most two "frontier spines" — one on each side of $[a : b)$ — because at every level there are at most two partially-overlapping nodes. Hence $|\mathcal{P}| \leq 2\log_2 n$, and each $w \in \mathcal{P}$ contributes at most one sibling to π, giving $|\pi| \leq 2\log_2 n$.

The total proof size is therefore $O(\log n)$ aggregates and $O(\log n)$ hashes, *independent of* $b - a$, in contrast to the $\Theta((b-a)\log n)$ cost of one Merkle inclusion proof per transaction. Verification performs $O(\log n)$ hash evaluations and $O(\log n)$ additions.

Problem 2

Consider *bitcoin backbone executions* with $T = 2^{\kappa-15}$, $f = 0.1, \delta = 0.6, q = 1000$ and a hash function with $\kappa = 256$ bits. Suppose that the probability of an execution of interest *not* being typical is at most

$$4 \cdot 10^{12} \cdot e^{-\epsilon^2 \cdot \lambda \cdot f/3} + 4 \cdot 10^{20} \cdot 2^{-\kappa}$$

Numerically calculate:

1. The probability of a succesful query p.
2. Your choice of n, t.
3. Your choice of the Chernoff error ϵ that respects the balancing inequality.
4. A Chernoff interval parameter λ that ensures typical executions occur with probability at least $1 - 2^{-128}$.
5. The chain growth parameters: The chain growth interval s and the chain velocity τ.
6. The common prefix parameter k.
7. The chain quality parameters: The chunk size ℓ and the chain quality μ.

1. $p = \frac{T}{2^\kappa} = \frac{2^{\kappa-15}}{2^\kappa} = 2^{-15}$
2. Solving $f = 1-(1-p)^{q(n-t)}$ for $(n-t)$ to obtain $n-t = \frac{1}{q}\cdot\frac{\log(1-f)}{\log(1-p)} \approx 3$ and $t < (1-\delta)(n-t) \Rightarrow t < \frac{2n}{7} \Rightarrow t < \frac{2(t+3)}{7} \Rightarrow t < 6/5$. So we can choose $t = 1, n = 4$.
3. The balancing inequality says that $3\epsilon + 3f \leq \delta$. So we can choose $\epsilon = 0.1$.
4. Setting $4\cdot 10^{12}\cdot e^{-\epsilon^2\cdot\lambda\cdot f/3} + 4\cdot 10^{20}\cdot 2^{-\kappa} = 2^{-128}$, with $f = \epsilon = 0.1$, we get approximately $\lambda = 352083$.
5. The interval is $s = \lambda = 352083$ and the minimum velocity is $\tau = (1-\epsilon)f = (1-0.1)\times 0.1 = 0.09$ blocks per unit of time.
6. The common prefix parameter is $k = 2\lambda f = 2\times 352083\times 0.10 = 70416$.
7. The chunk is $\ell = k = 70416$ and the minimum quality is $\mu = 1 - \frac{(1+\epsilon)(1-\delta)(\lambda+2)}{(1-\epsilon)\lambda}$, which gives $\mu = 0.511$.

Problem 3

Consider a bitcoin backbone execution in the usual synchronous lockstep model. Suppose that, at the beginning of the execution, the genesis block G already has two children B_1 and B_2 forming a fork of length 1. From this point onwards, the honest parties mine exclusively on top of B_1 while the adversary mines exclusively on top of B_2. The adversary withholds every block it produces and only intends to release them once it has built a private chain of k blocks extending B_2.

We are interested in the following event E_k:

> *Consider the rounds R_H and $R_\mathcal{A}$ at which the honest parties and the adversary mine a chain of k blocks, respectively. The event E_k occurs if $R_H \leq R_\mathcal{A}$, i.e. the adversary mined k blocks before or at the same round as the honest parties mined a chain of k blocks.*

1. Calculate the probability $\Pr[E_k]$.
2. Find the limit of $\Pr[\lim_{k\to\infty} E_k = 1]$, assuming honest majority.

Note: Honest parties extend their longest chain by one block every successful round, while the adversary just need k succesful queries in order to mine k blocks.

Let $\bar{E}_k$ be the complement of E_k, i.e. the honest parties mine k blocks first. Let R_H be a random variable denoting the round at which the honest parties mine their k-th block. Each round is an independent success for the honest parties with probability $f = 1-(1-p)^{q(n-t)}$, so R_H follows a negative binomial distribution:

$$\Pr[R_H = r] = \binom{r-1}{k-1} f^k (1-f)^{r-k}, \qquad r \geq k\,.$$

By round r, the adversary has made rqt queries, each succeeding independently with probability p. Let A_r be the event that the adversary has fewer than k successes by round r.

$$\Pr[A_r] = \sum_{i=0}^{k-1} \binom{rqt}{i} p^i (1-p)^{rqt-i}\,.$$

For $\bar{E}_k$ to occur, there must be a round r for which $R_H = r$ and A_r occurs. Therefore, since the honest and adversarial mining processes are independent,

$$\begin{aligned}\Pr[\bar{E}_k] &= \sum_{r=k}^{\infty} \Pr[R_H = r] \cdot \Pr[A_r] \\ &= \sum_{r=k}^{\infty} \binom{r-1}{k-1} f^k (1-f)^{r-k} \sum_{i=0}^{k-1} \binom{rqt}{i} p^i (1-p)^{rqt-i}\,.\end{aligned}$$

Hence,

$$\Pr[E_k] = 1 - \Pr[\bar{E}_k]\,.$$

We express each first-passage time as a sum of i.i.d. geometric waiting times and apply the strong law of large numbers.
Let $W_1, W_2, \ldots$ be i.i.d. geometric random variables with parameter f, where W_i denotes the number of rounds between the $(i-1)$-th and i-th honest success:

$$\Pr[W_i = m] = (1-f)^{m-1} f, \qquad m \geq 1, \qquad \mathbb{E}[W_i] = \frac{1}{f}\,.$$

Then $R_H = \sum_{i=1}^{k} W_i$, and by the strong law of large numbers

$$\lim_{k\to\infty} \frac{R_H}{k} = \lim_{k\to\infty} \frac{1}{k} \sum_{i=1}^{k} W_i = \frac{1}{f} \quad \text{almost surely.}$$

For the adversary, let $V_1, V_2, \ldots$ be i.i.d. geometric with parameter p (the per-query success probability), where V_j counts the queries between the $(j-1)$-th and j-th adversarial success, so that $\mathbb{E}[V_j] = 1/p$. The total number of queries needed for k adversarial successes is $Q_k = \sum_{j=1}^{k} V_j$, and the corresponding round is $R_{\mathcal{A}} = \lceil Q_k/(qt) \rceil$. By the strong law of large numbers applied to $\{V_j\}$,

$$\lim_{k\to\infty} \frac{Q_k}{k} = \frac{1}{p} \quad \text{almost surely.}$$

For every real x we have $x \leq \lceil x \rceil < x + 1$, so

$$\frac{Q_k}{kqt} \leq \frac{R_{\mathcal{A}}}{k} < \frac{Q_k}{kqt} + \frac{1}{k}\,.$$

Since $\lim_{k\to\infty} \frac{Q_k}{kqt} = \frac{1}{pqt}$ and $\lim_{k\to\infty} \frac{1}{k} = 0$ almost surely, by the squeeze theorem

$$\lim_{k\to\infty} \frac{R_{\mathcal{A}}}{k} = \frac{1}{pqt} \quad \text{almost surely.}$$

Set $\mu_H = 1/f$ and $\mu_{\mathcal{A}} = 1/(pqt)$. The honest-majority condition $f > pqt$ is equivalent to $\mu_H < \mu_{\mathcal{A}}$. Let $\varepsilon = (\mu_{\mathcal{A}} - \mu_H)/3 > 0$. By the definition of the limits $\lim_{k\to\infty} \frac{R_H}{k} = \mu_H$ and $\lim_{k\to\infty} \frac{R_{\mathcal{A}}}{k} = \mu_{\mathcal{A}}$,

$$\exists\, K_H \in \mathbb{N} \colon \forall k \geq K_H \colon \left| \frac{R_H}{k} - \mu_H \right| < \varepsilon\,,$$

$$\exists\, K_{\mathcal{A}} \in \mathbb{N} \colon \forall k \geq K_{\mathcal{A}} \colon \left| \frac{R_{\mathcal{A}}}{k} - \mu_{\mathcal{A}} \right| < \varepsilon\,.$$

Let $K = \max(K_H, K_{\mathcal{A}})$. Then $\forall k \geq K$,

$$\frac{R_H}{k} < \mu_H + \varepsilon = \mu_{\mathcal{A}} - 2\varepsilon < \mu_{\mathcal{A}} - \varepsilon < \frac{R_{\mathcal{A}}}{k}\,,$$

hence $R_H < R_{\mathcal{A}}$. hence $R_H < R_{\mathcal{A}}$. We have shown that the event $\{R_H < R_{\mathcal{A}}\}$ holds for all $k \geq K$ almost surely, so

$$\lim_{k\to\infty} \Pr[R_H < R_{\mathcal{A}}] = 1\,.$$

Since $\bar{E}_k = \{R_H < R_{\mathcal{A}}\}$, we conclude that

$$\Pr[\lim_{k\to\infty} E_k = 1] = 0\,.$$

References

Some helpful definitions are provided below. For the full definitions, consult the lecture notes and the bitcoin backbone paper.

Definition (The Proof-of-Work Inequality)**.**

$$H(B) < T$$

Definition (Chain Growth (formal))**.** An execution has *chain growth*, parametrized the growth interval s and the velocity τ if, for any rounds r_1, r_2 with $r_2 \geq r_1 + s$, and any honest party P, it holds that $|C^P_{r_2}| \geq |C^P_{r_1}| + s\tau$.

Definition (Common Prefix (formal))**.** An execution has *common prefix*, parametrized by $k \in \mathbb{N}$, if for all honest parties P_1, P_2 and for all times $r_1 \leq r_2$, it holds that $\mathcal{C}^{P_1}_{r_1}[:-k] \preceq \mathcal{C}^{P_2}_{r_2}$.

Definition (Chain Quality (formal))**.** An execution has *chain quality*, parametrized by the chunk ℓ and the quality μ if, for all honest parties P and round r, and for any positions $i < j$ in the chain with $j > i + \ell$, the ratio of honest to total blocks in $\mathcal{C}^P_r[i{:}j]$ is at least μ.

Definition (Honest Majority (formal))**.** An execution is said to satisfy *honest majority* with *honest advantage* δ if $t < (1-\delta)(n-t)$.

Definition (The Balancing Equation)**.** The balancing equation is given by

$$3\epsilon + 3f \leq \delta\,.$$

One possible configuration is $\epsilon = f = \frac{\delta}{6}$.

Definition (Typicality)**.** An execution is called *typical* if for all sets S of consecutive rounds, with $|S| \geq \lambda$, the random variables $X(S)$, $Y(S)$, $Z(S)$ are at most an ϵ error within their expectations:

- $(1-\epsilon)\mathbb{E}[X(S)] < X(S) < (1+\epsilon)\mathbb{E}[X(S)]$.
- $(1-\epsilon)\mathbb{E}[Y(S)] < Y(S) < (1+\epsilon)\mathbb{E}[Y(S)]$.
- $(1-\epsilon)\mathbb{E}[Z(S)] < Z(S) < (1+\epsilon)\mathbb{E}[Z(S)]$.

Definition (Chernoff Bound)**.** Consider a set of independent and identically distributed Bernoulli trials $\{X_i\}_{i\in[n]}$ with $\mathbb{E}[X_i] = p$. Consider their Binomial sum $X = \sum_{i=1}^{n} X_i$, and its expectation $\mu = \mathbb{E}[X] = np$. The probability of X deviating more than ϵ from $\mathbb{E}[X]$ is negligible in n. Concretely,

$$\Pr[X \leq (1-\epsilon)\mathbb{E}[X]] \leq e^{-\epsilon^2\mu/2}$$
$$\Pr[X \geq (1+\epsilon)\mathbb{E}[X]] \leq e^{-\epsilon^2\mu/3}\,.$$

Chain addressing notation.

- $\mathcal{C}_r^P$: The chain of party P at round r.
- $|\mathcal{C}|$: Chain length
- $\mathcal{C}[i]$: i^{th} block in the chain (0-based). The block height is i.
- $\mathcal{C}[-i]$: i^{th} block from the end.
- $\mathcal{C}[0]$: Genesis (by convention honest).
- $\mathcal{C}[-1]$: The tip.
- $\mathcal{C}[i{:}j]$: Chain chunk from block i (inclusive) to j (exclusive).
- $\mathcal{C}[{:}j]$: Chain chunk from the beginning and up to block j (exclusive).
- $\mathcal{C}[i{:}]$: Chain chunk from block i (inclusive) onwards.
- $\mathcal{C}[{:}-k]$: The stable chain.

Variables.

- κ: The security parameter, and size of the hash function output.
- H: The hash, modelled as a random oracle.
- n: The number of parties.
- t: The number of corrupt parties.
- δ: The honest advantage.
- q: Compute per party per round (number of allowed queries to the random oracle).

- T: The mining target.
- s: The chain growth interval, in units of time.
- τ: The chain velocity, in blocks per unit of time.
- k: The common prefix parameter, in blocks.
- ℓ: The chain chunk size required to ensure quality, in blocks.
- μ: The chain quality as a ratio.
- p: The probability of a successful query.
- f: The probability that a given round is successful.
- ϵ: The Chernoff error.
- λ: The Chernoff interval.
- X_r: The random variable indicating whether a round was successful.
- Y_r: The random variable indicating whether a round was a convergence opportunity.
- Z_r: The random variable counting adversarially successful queries in a round.
- $X(S)$: The random variable counting the number of successful rounds among rounds S.
- $Y(S)$: The random variable counting the number of convergence opportunities among rounds S.
- $Z(S)$: The random variable counting the number of adversarially successful queries among rounds S.