Theory Exercise 3

Due: 1:15pm, Thursday, Mar 16, 2023

Please solve all the problems below. The problems are worth equal points. After a genuine attempt to solve the homework problems by yourself, you are free to collaborate with your fellow students to find solutions to the theory homework problems. Regardless of whether you collaborate with other students, you are required to type up or write your own solutions. Copying homework solutions from another student or from existing solutions is a serious violation of the honor code. Please take advantage of the instructors' and TA's office hours. We are here to help you learn, and it never hurts to ask! The assignments should be submitted via Gradescope.

Problem 1

In class, we studied *binary* Merkle trees, but Merkle trees can have higher dimensions. A d-dimensional Merkle tree is a *complete* tree in which every non-leaf node has exactly d children. The value of a leaf is the hash of its contents, as in a binary Merkle tree. The value of an internal node is the hash of the concatenation of its children's values. For example, in a ternary Merkle tree constructed using the hash function H, an internal node with value v which has children with values v_1, v_2, and v_3 will take the value $v = H(v_1 \,\|\, v_2 \,\|\, v_3)$.

Consider a ternary tree constructed using `blake2s`, a hash function with an output of $\kappa = 256$ bits which has 3^7 leaves. Calculate the proof size, in bytes, for this Merkle tree. You can assume the verifier already has the Merkle tree root and has received the claimed contents and index of the leaf node in question, so you do not need to include these in your proof size calculation.

Problem 2

In class, we studied Merkle trees where the leaves are relatively few. Many modern blockchain applications make use of *sparse* Merkle trees. A *sparse* Merkle tree is a complete binary Merkle tree with exactly 2^κ children. Since the tree has an exponential number of children, they cannot be processed by a polynomially bound computer. However, the vast majority of these children have contents set to the empty string, whereas a polynomial number of leaves have non-empty contents. This makes computing the root and proofs within these trees efficiently possible.

Consider a *sparse* Merkle tree with 2^κ children constructed using `sha256`, a hash function with an output of $\kappa = 256$ in which the *first* leaf has contents equal to the string `hello` and value equal to the hash

   `2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824`.

All the other $2^{256} - 1$ leaves of this tree have their contents set to the empty string and their value is equal to

   `e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855`.

What is the value of the root of this tree?

Problem 3

Consider a *bitcoin backbone execution* with $n = 10$, $t = 2$, $q = 1000$, $\kappa = 256$ and $T = 2^{\kappa-7}$. Numerically calculate the following quantities:

1. The honest advantage δ.

2. The probability p of a successful query.

3. The probability f that a round is successful.

4. The expectation $\mathbb{E}[Z_r]$ of the number of successful adversarial queries in a round.

5. The expectation $\mathbb{E}[Z(S)]$ of the number of successful adversarial queries in a set of consecutive rounds with $|S| = 256$.

6. Your best lower bound for $\mathbb{E}[Y_r]$, the expectation of a round being a successful query.

Problem 4

Consider *bitcoin backbone executions* with $t = 2$, $n = 10$, $q = 1000$ and a hash function with $\kappa = 256$ bits. The probability of an execution being typical is at least

$$1 - e^{-(\epsilon^2 \lambda f + \kappa - 30)}.$$

Numerically calculate:

1. The honest advantage δ.

2. Your choice of a secure parametrization for the probability f of a successful round and the Chernoff error ϵ that respects the balancing equation.

3. The probability of a successful query p.

4. A suitable target T to achieve it.

5. A Chernoff interval parameter λ that ensures typical executions occur with probability at least $1 - 2^{-256}$.

6. The chain growth parameters: The chain growth interval s and the chain velocity τ.

7. The common prefix parameter k.

8. The chain quality parameters: The chunk size ℓ and the chain quality μ.

Problem 5

When we implemented the Selfish Mining simulation in class using the Monte Carlo method, we implicitly assumed that each block is instantly received by the whole network before the next block is mined. This allowed us to deduce that the probability of the next produced block being adversarial was $\frac{t}{n}$. However, when we explored the fan-out attack and formalized the bitcoin backbone model, we saw that time discretization into rounds leads to situations in which multiple successful queries can occur in a given round.

Consider a bitcoin backbone execution in the usual synchronous lockstep model. We call a round *adversarially superior* if the round contains an adversarially successful query, but is not a successful round (i.e., there is no honestly successful query). We are interested in the following event E:

An adversarially superior round has occurred *strictly prior* to the first successful round.

For example, if rounds 1, 2, and 3 contain no successful queries, round 4 contains one adversarially successful query, round 5 contains no successful queries, and round 6 contains one honestly successful query, then the event E has occurred. On the contrary, if the rounds 1, 2, and 3 contain no successful queries, and round 4 contains both an adversarial and an honestly successful query, the event E has not occurred, because the adversarially superior round was the same as the first successful round.

For simplicity, assume the execution continues for infinite, not just polynomial, duration.

1. Analytically calculate the probability of the event E occurring. Simplify your calculation to closed form (containing no summations $\sum$).

2. Use Bernoulli's approximation $(1 - x)^a \approx 1 - ax$ for large a and small x to simplify your expression.

3. Find the limit of your probability for $p \to 0$, i.e., successful queries become rarer and rarer.

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).

$$H(B) \leq T$$

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 $|C_{r_2}^P| \geq |C_{r_1}^P| + s\tau$.

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 $\mathcal{C}_{r_1}^{P_1}[:-k] \preceq \mathcal{C}_{r_2}^{P_2}$.

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 $\mathcal{C}_r^P[i{:}j]$ 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.