Private Proof-of-Stake Blockchains using Differentially-private Stake Distortion

Chenghong Wang[1], David Pujol[1], Kartik Nayak[1], and Ashwin Machanavajjhala[1]

Duke University
{chwang, david, kartik, ashwin}@cs.duke.edu

Abstract. Safety, liveness, and privacy are three critical properties for any private proof-of-stake (PoS) blockchain. However, prior work (SP'21) has shown that to obtain safety and liveness, a PoS blockchain must *in theory* forgo privacy. Specifically, to ensure safety and liveness, PoS blockchains elect parties based on stake proportion, potentially exposing a party's stake even with private transaction processing. In this work, we make two key contributions. First, we present the first stake inference attack applicable to both *deterministic* and *randomized* PoS with exponentially less running time in comparison with SOTA designs. Second, we use differentially private stake distortion to achieve privacy in PoS blockchains and design two stake distortion mechanisms that any PoS protocol can use. We further evaluate our proposed methods using Ethereum 2.0, a widely-recognized PoS blockchain in operation. Results demonstrate effective stake inference risk mitigation, reasonable privacy, and preservation of essential safety and liveness properties.

1 Introduction

Nakamoto introduced Bitcoin's protocol [NB08], promoting open, permissionless participation in distributed ledger maintenance. However, achieving consensus in the permissionless setting is hard as the traditional honest majority (or two-thirds majority) assumption no longer hold as the adversary can create any number of Sybil parties [Dou02] Permissionless protocols address safety against Sybils by leveraging constraint resources, assuming honest parties hold the majority of such resources. They are classified into Proof-of-Work (PoW) protocols [NB08,SCG$^+$14,VJR18], reliant on computing power, and Proof-of-Stake (PoS) protocols using monetary stake [CM16,Woo16,Cha21,KN12,BLMR14,BGM16,BPS16]. While both protocols mark a major breakthrough in distributed consensus in a permissionless context, the reliance on expensive computing power leads to significant energy wastage in PoW [OM14,CDE$^+$16], prompting a trend of shifting from PoW to PoS blockchains [SCG$^+$14,Cha21].

The open setup of the blockchain makes transaction privacy an important issue. Although, there are PoW designs [SCG$^+$14,NM$^+$16,BCG$^+$19] have taken the first step toward addressing this by concealing transaction details using cryptographic primitives while maintaining blockchain functionality. In PoS blockchains, merely hiding transactions is insufficient, as the stake is also sensitive [KMNS21]. Although some efforts [KKKZ19,GOT19,BMSZ20] address PoS privacy by hiding both transaction and stake, a recent study [KMNS21] theoretically shows that SOTA designs remain susceptible to side-channel leakages, i.e., the frequency a party adding a new block, which correlates with the party's stake. Additionally, they introduce reverse tagging attacks (RTA), enabling attackers to determine if a party's stake exceeds a specified value. The basic attack primitive has been deemed feasible in real-world blockchains like Zcash [SCG$^+$14]. Kohlweiss et al. [KMNS21] also propose a proof-of-concept stake inference attack (SIA) that estimates a party's unknown stake by repeatedly using RTA with varying comparison values.

This paper tackles the following open questions. The stake inference attack in [KMNS21] is impractical due to simplifying assumptions like deterministic protocols and strict liveness guarantees. These assumptions do not hold in real-world executions as PoS blockchains typically operate under a probabilistic manner [CM16,Cha21]. Hence, the first open question is whether there are stake inference attacks that can be launched in a real-world context. Moreover, we are not aware of any PoS design that can ensure privacy against attacks like RTA, or the derived SIA. Thus, the second open question is if there exist private PoS designs that address SIA risks. Motivated by the open questions, we elaborate on our contributions as follows:

1.1 Contributions

C-1. Stake inference attacks (SIA) for PoS protocols.

– *A general stake inference paradigm.* We propose a general stake inference paradigm that provides a template for innovating efficient and accurate SIA that can be run in practice. Our paradigm captures randomized protocols by formulating liveness in a probabilistic manner and considers robust inference strategies that tolerate unreliable feedback, i.e. incorrect comparisons between a candidate value and the target stake.

– RdBin, *a concrete stake inference attack.* Following our inference paradigm, we provide RdBin, the first practical SIA against randomized PoS protocols. RdBin combines random walk and binary search to adaptively approximate the target stake with exponentially more efficient running time than the SOTA one [KMNS21], i.e. sublinear complexity related to the total stake S in contrast to the linear time ensured by [KMNS21].

C-2. Private PoS using DP Stake Distortion.

– *Formulating private PoS security model.* In the presence of SIAs, existing PoS protocols under the standard UC security model [KKKZ19,KMNS21] are clearly inadequate to achieve privacy. To address this, we define two desiderata that ensures a practical PoS blockchain has rigorous and well-defined privacy. Based on this, we take the first attempt to integrate differential privacy (DP) [DR+14] with previous UC security models of private ledgers and derive a formal definition for the private PoS ledger. The improved model inherits key components of UC private ledger definitions but also formulates provable stake and transaction privacy.
– *Two stake distortion mechanisms that extend standard PoS to private PoS while balancing liveliness, safety and privacy.* Though naïve methods exist, i.e. partition (disconnect) each party or use equal chance for leader election, that address the potential privacy violations (SIA risks). The former completely abandons liveness while the latter provides no safety. In this work, we propose two DP stake distortion mechanisms, namely Timer and Binary mechanism, which can be mounted to any PoS protocol and extend the ordinary PoS into a private PoS under our security model. Moreover, the resulting protocols ensure provable stake and transaction privacy while preserving the original safety and liveness guarantees at the same time.
– *Publicly verifiable protocols that securely realize DP distortion mechanisms.* In addition to the idealized functionalities, we construct concrete protocols that securely realize the them. To the best of our knowledge, the protocols are the first attempts to provide publicly verifiable DP mechanisms over continual observations.
– *Case study with real-world blockchains* We conduct case studies on Ethereum 2.0 [Cha21] to show that the stake distortion imbues real-world blockchains with reasonable stake privacy while retaining the original safety and liveness. We also compare the methods and examine how they differ in ensuring these properties.
– *Prototype protocol and efficiency evaluation.* We implement prototype stake distortion protocols and evaluate their efficiencies. The result indicates evaluating stake distortions adds small overheads to blockchain protocols (i.e. less than 13% of Zcash [SCG+14]'s protocol). Moreover, the distortion occurs over certain intervals, which causes the amortized cost to be even smaller.

The paper is structured as follows: Section 2 covers background, Section 3 addresses stake inference attacks, Sections 4 and 5 present private PoS desiderata and our privacy model, while Sections 6,7, and8 detail our private PoS design, case studies, and evaluations.

2 Background

Private ledger. We now examine the private ledger functionality (2.1), $\mathcal{G}_{\mathsf{PL}}$, considered by this work, which is adapted from earlier UC formulations [KKKZ19,KMNS21] of private ledgers. Given its complexity, we outline the functionality concisely, suitable for technical dialogues, with extended analysis in Appendix A.

Functionality 2.1: $\mathcal{G}_{\mathsf{PL}}$

The functionality $\mathcal{G}_{\mathsf{PL}}$ manages a general ledger state, state; a local state state_i (a prefix of state) for every party P_i; a buffer of unconfirmed transactions, buffer; and a sequence of honest inputs $\mathcal{I}_H$.

Upon receiving any input I from party P_i, record $\mathcal{I}_H \leftarrow \mathcal{I}_H || (I, P_i, t)$, if I is not `SUBMIT` command, then evaluate the following:

(a) (Add Transaction). If I is command $(\mathtt{SUBMIT}, \mathsf{sid}, \mathsf{tx})$, and tx is valid. Add tx to buffer, and update $\mathcal{I}_H = \mathcal{I}_H || (I, \mathsf{blind}(\mathsf{tx}), t)$ with the blinded transaction, for instance, $\mathsf{blind}(\mathsf{tx})$ hides the sender, recipient, and the amount of tx

(b) (Read State). Return the blinded version of P_i's local state, $\mathsf{blind}(\mathsf{state}_i)$, with all non-$P_i$ transactions replaced by blind ones. If P_i is corrupted, send $(\mathsf{blind}(\mathsf{state}_i), \mathcal{I}_H, \mathsf{Lkg})$ to the adversary.

(c) (Extend chain). If I is $(\mathtt{MAINTAIN\_LEDGER}, \mathsf{sid})$, then perform the *ledger maintenance* [KRDO17], [DGKR18], [BGK+18] to add valid transactions in buffer to state.

In line with prior formulations [KKKZ19,KMNS21], we let the adversary know when corrupted party transactions are confirmed. Thus, in $\mathsf{blind}(\mathsf{state}_i)$, only non-$P_i$ transactions are replaced with blind ones. In addition, $\mathcal{G}_{\mathsf{PL}}$

is also parameterized with a leakage profile Lkg, revealing block proposer (the party who maintains the ledger) identities per round. It is noteworthy that Kohlweiss et al. [KMNS21] theoretically demonstrated the nonexistence of private ledgers without leaking block proposer identities (even in presence of anonymous broadcast channels). We stress that Lkg is indeed protocol-specific which especially depends on how the protocol elects the block proposer. However, in PoS blockchains, block proposers are typically chosen through a "private lottery" [GOT19], where the odds of a party winning ledger maintenance eligibility is proportional to their stake. Hence, one may nevertheless formulate Lkg as a (possibly probabilistic) function over each party's stake (See Appendix A.3 for an example leakage profile).

Definition 1. *Protocol $\prod$ is said to be a private ledger with leakage Lkg if the ptotocol UC-realizes [Can01] functionality $\mathcal{G}_{\mathsf{PL}}$ in the presence of a p.p.t. adversary.*

Private ledger properties. Following the private distributed ledger properties discussed in previous works [KRDO17,DGKR18,KKKZ19,KMNS21], we formulate important properties considered by this work.

Definition 2 (Private ledger properties). *Let $\prod$ to be a private ledger protocol executed by a set of parties. We consider the following properties hold when f fraction of the parties is malicious.*

– **Safety***: No two honest users disagree on confirmed transactions (i.e., transactions appended to* state*).*
– (z,t)**-Liveness***: Any valid transaction that is input to at least an honest $z \leq (1-f)$ fraction of the total parties, will be appended to* state *within t time steps.*

Consistent with [CM16,Buc16,Maz15,XLCB18], we consider at least $\frac{2}{3}$ fraction of the total stake is owned by the honest parties, and correspondingly $f < \frac{1}{3}$ is interpreted as malicious corrupted stake fraction. Moreover, for various reasons, the properties in Definition 2 may be ensured in a probabilistic manner. For example, Algorand [CM16] and Ouroboros [KRDO17] use randomness for efficiency thus ensures safety or liveness only with high probability.

Privacy tradeoffs and attacks against PoS. Kohlweiss et al. [KMNS21] demonstrated that it is theoretically impossible to have a private ledger ensuring liveness without revealing the block proposer's identity. Moreover, recognizing block proposers over time reveals the frequency at which parties take on the block proposer role. In PoS blockchains, such frequency correlates with a party's stake, potentially leading to stake privacy loss and allowing attackers to deduce an individual's stake. For example, Kohlweiss et al. [KMNS21] examined the Reverse Tagging Attack (RTA), enabling attackers to assess if a party's stake surpasses a specified threshold. RTA's idea involves creating input disparities between a target and other nodes, such as by delaying specific transactions. For example, the attacker delays a self-created transaction tx for a target P_v with stake f_v, and a set of corrupted nodes with stake f_{cmp}. If tx isn't confirmed after t steps, by (z,t)-liveness, the attacker learns $f_v \geq 1-z-f_{\mathsf{cmp}}$.

The delay operation was implemented using the *Invblock* technique, first proposed in [MLP+15] and was later used to infer Bitcoin topology [DSBPS+19]. Kohlweiss et al. [KMNS21] further assessed the validity of *Invblock* against private ledgers, i.e. Zcash [SCG+14]. Note that, the cost of launching *Invblock* is low, as it necessitates the attacker becoming one of the victim's peers in the P2P network, rather than compromising all target node peers.

Additionally, [KMNS21] also outlined a stake inference attack (SIA) design by iteratively applying RTA with adaptive f_{cmp}. The attacker starts with $f_{\mathsf{cmp}} = 0$ and repeats RTA, increasing f_{cmp} by $\frac{1}{S}$ each time until the first f_{cmp} satisfies $f_v \geq 1-z-f_{\mathsf{cmp}}$, where $\hat{f}_v = 1-z-f_{\mathsf{cmp}}$ approximates f_v.

Differential privacy [DR+14]. guarantees that altering one input data element in an algorithm or mechanism imposes minimal impact on the output. More specifically, let D and D' be two databases that differ by only a single tuple, namely neighboring databases, then:

Definition 3 (ϵ-differential privacy). *Given $\epsilon > 0$, a randomized mechanism $\mathcal{M}$ is ϵ-DP if for all pair of neighboring databases D, D', and any possible output $o \subset Range(\mathcal{M})$, the following holds:*

$$\Pr\left[\mathcal{M}(D) \in o\right] \leq e^{\epsilon} \Pr\left[\mathcal{M}(D') \in o\right]$$

An alternative definition, known as *approximate DP* or (ϵ,δ)-DP, allows a small failure probability δ in addition to the constraints in Definition 3. In this paper, the focus is on ϵ-DP, while an extended discussion that considers the relaxed (ϵ,δ)-DP can be found in Appendix E.

3 Stake Inference Attack

In this section, we discuss practical stake inference attacks against PoS protocols. In general, given a target party P_v with stake f_v, we consider attack strategies to adaptively choose $\hat{f}_v$ that minimizes the error $|\hat{f}_v - f_v|$.

3.1 Issues with the SOTA Approach [KMNS21]

I-1. Restricted to deterministic protocols. Kohlweiss et al. [KMNS21] consider rather idealized deterministic protocols, assuming stringent liveness conditions. However, this assumption may not align with real-world scenarios, as most blockchains function probabilistically, with liveness typically guaranteed at a high probability [CM16,Cha21].

I-2. RTA: One-sided comparison only. Based on (z, t)-liveness, tx can be confirmed even if less than z fraction of honest parties receive it. Therefore, the $s_v + f_{\mathsf{cmp}} < (1-z)$ conclusion may not hold when tx is confirmed within t steps, potentially causing f_v to be larger than $\hat{f}_v$. This one-sided comparison limitation makes RTA-based SIA less reliable for accurate stake inferences.

I-3. Inefficient search strategy. The linear scan strategy considered by [KMNS21] requires a large number of RTA executions, which is inherently inefficient.

The aforementioned limitations reduce the practicality of RTA-based SIA. In what follows, we focus on developing a practical SIA that is efficient, accurate, and suitable for randomized PoS protocols.

3.2 The General Stake Inference Paradigm

In this section, we show that the problem of finding accurate stake inference against probabilistic PoS blockchains can be reduced to a variant of *Noisy Search* problem [KK07].

Definition 4 (Noisy search problem). *Given n coins sorted by head probabilities (p_i for the i^{th} coin) and a target coin with $p_1 \leq p^* \leq p_n$, an algorithm must find two coins $i, i+1$ satisfying a given $\tau > 0$ such that $[p_i, p_{i+1}]$ intersects $[p^*-\tau, p^*+\tau]$, without knowing exact head probabilities but being allowed to flip coins*

The general reduction idea is to map stake to simulated biased coins so that inferring stake values reduces to locating search coins that is close to a target coin with heads probability tied to f_v. Details are as follows.

Formulating liveness. To simulate stake-based biased coins, we rely on the liveness ensured by the blockchain. To apply to randomized protocols, we adapt the (z, t)-Liveness definition accordingly.

Definition 5 (Probabilistic (z, t)-liveness). *For any transaction* tx *that is input to z fraction of the honest parties, the probability* tx *is NOT confirmed after t time steps equals to $q(1-z, t)$, where $q(\cdot)$ is a monotonically increasing function related to* $1-z$.

In general, $q(\cdot)$ is highly related to the protocol specification. For example, assuming the probability of a party proposing the next block is determined by a slot function $\phi(\cdot)$ [GOT19,KKKZ19,KMNS21] over its stake. One may obtain $q(1-z, t) = (1-\phi(z))^t$ for any $z, t > 0$. Although, this probability may differ from the theoretical one in real executions, i.e., due to some other configurations, such as timeouts, transaction pool size, etc. One may assume that as long as the protocol specification does not change, the value of $q(z, t)$ remains stable for any z and t. Moreover, the actual value of $q(\cdot)$ in real executions may be not directly computable, but one may evaluate it empirically.

Simulating "stake-based" biased coins. Given a node (or a group of nodes) with relative stake f_x, we provide an interface for simulating a biased coin with heads probability equal to $q(f_x, t)$ for any $t > 0$. Note there are two types of biased coins, the search coin and the target coin, for which the simulations are different. In simulating the search coin, the attacker knows the pre-image f_x, while it is unknown when simulating the target coin.

To simulate a search coin, the attacker first samples a set of corrupted nodes with total stake f_x, then broadcasts a self-created transaction, tx, to everyone, but removes tx from the sampled nodes' transaction pool. The attacker waits for t time steps to observe if tx is confirmed. By Definition 5 the probability of tx not being confirmed after t steps is $q(f_x, t)$. For the target coin, we utilize the delay operation considered by previous works [KMNS21,MLP$^+$15]. Specifically, the attacker broadcasts tx to everyone but delays it to the victim, such that tx does not reach P_v for at least t time steps. Similarly, the attacker then waits to observe if tx is confirmed after t steps, where the "not confirmed" probability equals $q(f_v, t)$.

For demonstration purposes, we consider $t = 1$ to be the default setting and therefore omit it. In what follows, we use p_x to directly denote the heads probability of a stake-based biased coin with pre-image f_x.

Reduction. We provide an abstract algorithm for SIA by assuming the existence of a black-box solver, `ns_solver`, for the noisy search problem. Given P_v with stake f_v, and n sorted stake values, the algorithm simulates corresponding search coins as well as the target coin then inputs them to `ns_solver` along with a specified τ. Upon receiving two coins $i, i+1$ output from `ns_solver`, the algorithm computes $q^{-1}(p_i)$ and, $q^{-1}(p_{i+1})$ to obtain the pre-images f_a, and f_b, then returns $\frac{f_a+f_b}{2}$ as $\hat{f}$. Note that, in simulating the search coins, the algorithm knows the pre-image of each stake-based biased coin, and thus $\frac{f_a+f_b}{2}$ can be obtained within constant time. Therefore, if `ns_solver` requires $T(n)$ times coins tosses before termination, then the running time of the SIA is dominated by $T(n)$ many simulated coin flips. Figure 1 shows a diagram of the aforementioned reduction.

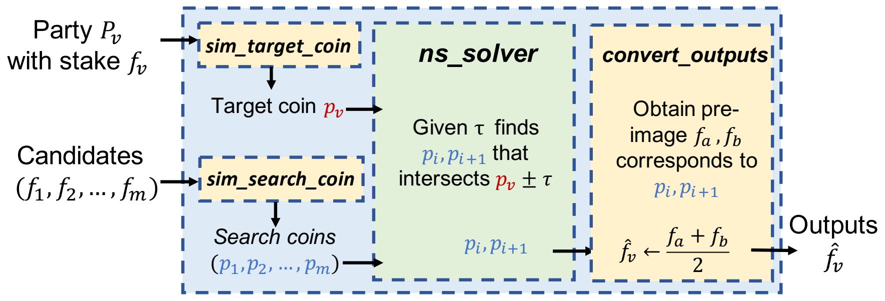

Party P_v with stake f_v
Candidates $(f_1, f_2, \dots, f_m)$
sim_target_coin
Target coin p_v
sim_search_coin
Search coins $(p_1, p_2, \dots, p_m)$
ns_solver
Given τ finds p_i, p_{i+1} that intersects $p_v \pm \tau$
p_i, p_{i+1}
convert_outputs
Obtain pre-image f_a, f_b corresponds to p_i, p_{i+1}
$\hat{f}_v \leftarrow \frac{f_a+f_b}{2}$
Outputs $\hat{f}_v$

Fig. 1: Reduction diagram.

3.3 Practical Stake Inference Attack

Following our reduction, we provide RdBin, a practical SIA that combines random walk and binary search. Similar to [KMNS21], we examine a loosely dynamic scenario where honest parties' stakes remain stable over short periods but change over longer durations. RdBin is parametered by $\theta > 0$, and $\tau \in (0, 1)$, and is given a balanced (stake) binary tree ST with each node labeled with a stake segment $[f_a, f_b] \subseteq [0, f]$. Specifically, the root is labeled with $[0, f]$, and for every internal node v with label $[f_a, f_b]$, its left and right child is labeled with $[f_a, m]$ and $[m, f_b]$, respectively, where $m = \lfloor \frac{f_a+f_b}{2} \rfloor$. Moreover, all leaf nodes satisfy $f_b - f_a = \frac{\theta}{S}$. The algorithm starts with a random walk in ST from the root, and for each round t (assuming at node $v(t)$), it evaluates a probabilistic stake comparator, pSC, that compares f_v with the pivot of $v(t)$, say $\frac{f_a+f_b}{2}$. If $f_v < \frac{f_a+f_b}{2}$ is claimed by pSC, it moves to the left child, otherwise to the right child. RdBin halts and outputs $\hat{f} \leftarrow \frac{f_a+f_b}{2}$, if it reaches a leaf node or pSC asserts min_diff. The details are shown in Algorithm 1.

Algorithm 1 $p\mathsf{SC}(f_v, f_{\mathsf{cmp}}, \tau, \delta)$

```
1:  Sample a set of corrupted users $P_c$ with stake $f_{\mathsf{cmp}}$
2:  for $i = 1, 2, 3...$ do $p_0 = p_1 = 0$
3:      $\tau_i = e^{-i/2}$, $\delta_i = \delta/e^i$, $n_i = \log(1/\delta_i)/\tau_i^2$
4:      if $\tau_i < \tau$ then assert min_diff
5:      for $j = 1, ..., n_i$ do $(b \in \{0, 1\})$
6:          Broadcast conflicting transactions† $\mathsf{tx}_0$, $\mathsf{tx}_1$
7:          Delay $\mathsf{tx}_0$ to $P_v$ and remove $\mathsf{tx}_1$ from $P_c$.
8:          $p_b = p_b + 1/n_i$, if $\mathsf{tx}_b$ is not confirmed
9:      if $|p_0 - p_1| \geq 2\tau_i$ then
10:         assert $f_v > f_{\mathsf{cmp}}$ if $p_0 > p_1$ else $f_v \leq f_{\mathsf{cmp}}$
```

†As long as one transaction is confirmed, the other one is invalid, this allow us to flip two biased coins at the same time.

Theorem 6. *Given δ, $\tau \in (0, 1)$, $\tau_c = |q(f_v) - q(f_{\mathsf{cmp}})|$, and $\tau_m = \max(\tau, \tau_c)$, with probability at least $1-\delta$, pSC outputs correctly after expected $O\left(\tau_m^{-2} \ln(1/\delta\tau_m)\right)$ many simulated coin tosses.*

For each round i, the failure probability of testing coins is at most δ_i, thus by union bound [GS97], the failure probability for the entire testing is bounded by $\sum_i \delta_i < \delta$. By Hoeffding's inequality, the probability that pSC to continue running for $\tau_i < \frac{1}{2}\tau$ decreases exponentially in n_i. Thus the expected complexity is dominated by the complexity of round $i = \log(\frac{1}{\max(\tau, \tau_c)})$, which is consistent with Theorem 6. We extend this to full proofs in Appendix C.1. Overall, pSC provides accurate two-sided comparisons with a bounded failure probability. Moreover, by setting $\delta = (\log \frac{S}{\theta})^{-1}$, the random walk ensures movement towards the correct node with probability at least $1-(\log \frac{S}{\theta})^{-1}$, and by union bound, the failure probability of RdBin is bounded by a constant factor.

Theorem 7. *Let $n = \frac{S}{\theta}$, $\delta = O(1/\log n)$, $p_v = q(f_v)$, and $\eta = \max\left(|q^{-1}(p_v \pm \tau) - f_v|\right)$, then* RdBin *satisfies:*

1. *The running time is* $O(\tau^{-2}\log(\log n/\tau)\log n)$.
2. *With constant failure probability,* RdBin *outputs an inference* $\hat{f}_v$ *with error* $|\hat{f}_v - f_v| \leq \max(\frac{\theta}{S}, \eta)$.

Note that RdBin represents only one construction under our general inference paradigm, and there exist many other designs in this space. We continue in Appendix D by providing two more SIAs, each adapted from a classical noisy search algorithm. Due to the simplicity of RdBin and the fact that its implementation does not require extra memory, we present it as the default attack. One may also derive certain extensions on RdBin. First, it is possible to flip even more simulated coins at once. For instance, one can create a set of (conflicting) transactions, $\mathsf{tx}_1, \mathsf{tx}_2, ..., \mathsf{tx}_n$, broadcasting them all at once, but delaying the arrival of each tx_j to a specific target P_j. This further enables the attacker to construct a parallelized RdBin against multiple victims. Second, as the stake is the additive outcome of a series of transactions. It's obvious that one can utilize RdBin to extrapolate transactions outcome for any victim P_v between any two time steps.

4 Private PoS Desiderata

We discuss requirements for achieving a private PoS by defining two desiderata that are designed to ensure a practical protocol has rigorous and well-defined privacy.

D-1: Well-defined privacy on leakage. Any private PoS should provide a rigorous and provable bound on the leakage of each party's stake and transactions. To facilitate this we say that in addition to the standard UC security model (Definition 1), a private PoS should also meet additional privacy definitions, i.e., Definition 12, which constrains the knowledge that an adversary can obtain regarding each party's stake and transactions by observing the associated leakages.

D-2. Balance privacy, safety and liveness. While safety and liveness are traditionally ensured through a PoS protocol, introducing privacy may create tension between these properties. For instance, one could ensure privacy by using a consensus protocol that is completely independent of the parties' stake. But such a system would not be safe, since the fundamental purpose of introducing stake is to address safety against Sybils. Thus, given all of safety, liveness, and privacy may not be achievable simultaneously, we require a practical private PoS ledger should balance between these three properties.

5 Privacy Model

5.1 Privacy Requirements

PR-1. Limited stake inference. In general, we require that any attacker should not infer the current stake owned by a certain party (by observing the protocol leakage) within an additive bound α. We assume that the adversary knows all parties involved in the protocol, as their participation can be easily identified, for example, when block proposers broadcast the updated ledger states, it inevitably indicates their involvement in the ledger maintenance. Hence, we do not require stronger privacy in the form of "the existence of a party cannot be inferred".

PR-2. Privacy with expiration. Considering that the parties' stake changes dynamically, stake privacy should be ensured under continual observation [DNPR10]. Ideally, we desire to obtain the same degree of privacy for each time step based on all information disclosed until that time step. However, providing such strong privacy guarantees inherently leads to errors accumulating over time [DNPR10,Dwo10]. In the context of PoS blockchains, this can result in a large distorted adversarial stake potentially causing a significant decline in safety or liveness guarantees. Therefore, to balance between these properties, in this work, we adopt a relaxed "privacy with expiration" model [BFM+13,KPXP14,LSV+19a]. In this model, we aim to achieve strong privacy for parties' stake within a recent time frame while allowing the privacy of parties' past stake to gradually decay over time.

PR-3. Transaction privacy. We also consider transaction privacy, that is, we bound the attacker's ability to infer the outcome of any party's transaction. Typically, as the stake value directly reflects the additive outcome of transactions, achieving PR-1 and PR-2 also implies transaction privacy if the protocol leakage is subject to parties' stake, i.e., no information relevant to the transaction outcome is disclosed to the adversary, other than what can be obtained from the parties' stake at each time.

Putting it all together. Combining PR-1, 2, 3, we formulate our privacy requirements using Pufferfish [KM14] framework, a well-known tool that translates common privacy requirements into formal definitions [HMA+17].

Definition 8 (Stake privacy requirement). *Let* $P = \{P_1, P_2, ..., P_n\}$ *denote the set of participants with* $S^i = \{s^i_1, s^i_2, ...\}$ *as the stake history for party* P_i*, where* s^i_j *is* P_i*'s stake at time* j*. We denote* $\theta_i \in \Theta$ *as the attacker's* *belief about P_i's stake, where Θ is the set of all possible adversarial beliefs. Given a private ledger with leakage* Lkg, *and $\epsilon, \alpha > 0$, we require that at any time t the following holds for all $\theta_i \in \Theta$, and $o \in Range(\mathsf{Lkg})$.*

$$\frac{\Pr_{\theta_i}\left[s^i_j = x \mid \mathsf{Lkg} = o\right]}{\Pr_{\theta_i}\left[s^i_j = y \mid \mathsf{Lkg} = o\right]} \Bigg/ \frac{\Pr_{\theta_i}\left[s^i_j = x\right]}{\Pr_{\theta_i}\left[s^i_j = y\right]} \leq e^{\omega(t-j)\epsilon} \tag{1}$$

where $j \in [1,t]$ and $x, y \in \mathbb{R}$ such that $\Pr[s^i_j = x] \geq 0$, $\Pr[s^i_j = y] \geq 0$, and $x \leq y \leq x + \alpha$. ω is the privacy decay multiplier, which is a monotonically increasing function related to $(t-j)$ with $\omega(0) = 1$.

As in Definition 8, the privacy requirement limits the maximum Bayes factor that an attacker can learn, after observing the leakage Lkg, regarding the stake owned by any party at any point of time. Such a factor with respect to the most recent stake s_t is bounded by e^{ϵ}. Furthermore, for any stake s_j that is $(t-j)$ time steps away from current time, such a factor is bounded by $e^{\omega(t-j)\times\epsilon}$, which captures the notion of privacy decay (PR-2). Note that, Definition 8 also addresses transaction privacy (PR-3). More specifically, let $\mathsf{tx}_{j:j+1}$ as P_i's additive transactions outcome between time j and $j+1$. Since $\mathsf{tx}_{j:j+1} = s_{j+1} - s_j$ and by Equation 1, the attacker's posterior odds (after observing the leakage Lkg) about $\mathsf{tx}_{j:j+1} = x$ rather than $\mathsf{tx}_{j:j+1} = y$ $(x \leq y \leq x+\alpha)$ is at most $e^{\omega(t-j)\times\epsilon}$ times the attacker's prior odds.

5.2 Formal Privacy Definition

In this section, we discuss the formal privacy model considered by this work. First, we formulate the leakage privacy following the notion of differential privacy [DR+14].

Definition 9 **((α, u)-neighbors).** *Let $S_t = \{s_1, ..., s_t\}$ and $S'_t = \{s'_1, ..., s'_t\}$ be the any two stake histories up to time t. S_t and S'_t are (α, u)-neighbors if the following holds: (i) $\alpha > 0$, and $u \in (0, t]$; (ii) $s_j = s'_j$ for all $j \leq u$; and (iii) $s_j \leq s'_j \leq s_j + \alpha$ for all $j > u$.*

Definition 10 **((α, ϵ)-private leakage).** *Given* Lkg *that depends on stake,* Lkg *is said to be (α, ϵ)-private, if for any two (α, u)-neighbor stake histories S_t, and S'_t, any output $o \subset Range(\mathsf{Lkg})$, and any P_i, the following holds*

$$\Pr\left[\mathsf{Lkg}^{\langle P_i, S_t\rangle} = o\right] \leq e^{\omega(t-u)\epsilon} \cdot \Pr\left[\mathsf{Lkg}^{\langle P_i, S'_t\rangle} = o\right] \tag{2}$$

where $\mathsf{Lkg}^{\langle P_i, S_t\rangle}$ (resp. $\mathsf{Lkg}^{\langle P_i, S'_t\rangle}$) denotes the leakage when P_i's stake history is activated by S_t (resp. S'_t), and ω is the decay function related to $t-u$ with $\omega(0) = 1$.

Theorem 11. *If* Lkg *is (α, ϵ) private (Definition 10), then* Lkg *satisfies all privacy requirements (Definition 8).*

We stress that DP is a special case of Pufferfish privacy [KM14], and applying DP on party's stake history ensures all privacy requirements listed in Section 5.1. For complete proofs, interested readers can refer to Appendix C.2 Next, we formally define the private PoS ledger under the UC framework [Can01].

Definition 12 **((α, ϵ)-private PoS ledger).** *A PoS ledger protocol is said to be (α, ϵ)-private (resp. $(\alpha, \epsilon, \delta)$-private), if: (i) it UC realizes [BMTZ17] $\mathcal{G}_{\mathsf{PL}}$ with leakage* Lkg *in the presence of a p.p.t. adversary $\mathcal{A}$ and (ii) the leakage* Lkg *is (α, ϵ)-private (resp. $(\alpha, \epsilon, \delta)$-private).*

We stress that private ledger protocols are allowed to interact with auxiliary functionalities (hybrids), which captured the resources that are available to parties. The composability property [BMTZ17] of UC states that if π interacts with $\mathcal{F}$ to UC realize functionality $\mathcal{G}$, and $\pi_{\mathcal{F}}$ UC realizes $\mathcal{F}$, then substituting calls to $\mathcal{F}$ in π with calls to $\pi_{\mathcal{F}}$ results in a secure protocol for $\mathcal{G}$ in the hybrid world. This also provides the flexibility to analyze the security of complex protocols in a modular manner. In this paper, we primarily focus on the ledger maintenance protocol (i.e., the sub-protocol that UC realizes `MANTAIN_LEDGER` command in $\mathcal{G}_{\mathsf{PL}}$), as this is the main component that leaks stake-related information [DGKR18,GOT19,KKKZ19].

6 Private PoS with Stake Distortion

To mitigate SIA risks, naïve approaches exist. For instance, one may consider each party to maintain the ledger with the same chance, but clearly, this provides no safety at all. Although this can be improved by letting one party own multiple nodes, each contributing independently to ledger maintenance, it requires a single party running the private lottery protocol multiple times, causing significant overhead for large stakeholders. Another option is to disconnect or impose long delays for each party to produce new blocks, but this severely impacts liveness. Steered by our privacy model, we propose our private PoS design, which not only ensures provable privacy but also strikes a balance between safety and liveness.

6.1 Design Overview

We first provide a brief overview of the general design pattern (Protocol 6.1) for a PoS-based ledger maintenance protocol, from the view of a party P. This pattern is abstracted from previous constructions [LSV+19b,KMNS21,GOT19].

Protocol 6.1: Ledger Maintenance, $\prod_{\mathsf{LM}}$

```
1: Interact with functionality $\mathcal{F}_{\mathsf{stk}}$ to retrieve lottery inputs: $(\mathsf{in}, \mathsf{com}_{\mathsf{in}}, r) \leftarrow \mathcal{F}_{\mathsf{stk}}$
   // Lottery input in obtained from $\mathcal{F}_{\mathsf{stk}}$ is the party's true stake
   [Our modifications: We consider a modified ledger maintenance protocol $\prod_{\mathsf{LM}^*}$, where lottery inputs
   are obtained via interacting with $(\mathsf{in}, \mathsf{com}_{\mathsf{in}}, r) \leftarrow \mathcal{F}_{\mathsf{sd}}$, and in returned by $\mathcal{F}_{\mathsf{sd}}$ is the distorted stake]
2: Evaluate a private lottery: $\mathsf{ret} \leftarrow \mathsf{priv\_lottery}(\mathsf{in})$
3: if $\mathsf{ret} == \mathtt{WIN}$ then
4:     Generate zero-knowledge proof, $\pi$, proving:
         (a) the party wins the lottery with input $\mathsf{in}$
         (b) lottery input $\mathsf{in}$ is consistent with $\mathsf{com}_{\mathsf{in}}$.
5:     Perform the ledger maintenance activities, i.e. propose a new block or take the part of a committee
   to determine the next block.
```

In general, ledger maintenance execution occurs in discrete time slots, during which slot leaders are elected through private lotteries to update ledger states. Specifically, in each round, each party P interacts with a functionality $\mathcal{F}_{\mathsf{stk}}$ [GOT19] to obtain the lottery input in and partakes in a private lottery with winning odds tied to in. If P wins, then she contributes to ledger maintenance along with a zero-knowledge proof of eligibility. Typically, to implement such a proof, other parties should also access a hiding commitment $\mathsf{com}_{\mathsf{in}}$ to P's lottery input from $\mathcal{F}_{\mathsf{stk}}$ (while in practice, such commitments can efficiently be computed from the blockchain[KKKZ19]). However, the actual value of in and the opening r for $\mathsf{com}_{\mathsf{in}}$ remain exclusive to P. In a PoS blockchain, the lottery input in returned by $\mathcal{F}_{\mathsf{stk}}$ is P's **true stake** at the current time.

However, as mentioned before, the standard PoS design unavoidably reveals the frequency of each party's lottery wins, closely linked to the party's true stake and enabling accurate stake inferences. To mitigate this, our design modifies how parties acquire lottery inputs while imposing no changes to other components. Specifically, we consider parties interacting with a new functionality, $\mathcal{F}_{\mathsf{sd}}$, which produces a **distorted stake** as the lottery input for each party at every time.

This suggests that one may directly derive an actual protocol under our design from the standard one by mounting functionality $\mathcal{F}_{\mathsf{sd}}$, i.e., replacing how the parties obtain private lottery inputs with a subroutine call to $\mathcal{F}_{\mathsf{sd}}$. As the input retrieval is completely independent of the core protocol, the derived protocol thus retains the security guarantees of the original protocol [GOT19]. Moreover, the leakage for the derived protocol is no longer related to users' true stake, but rather tied to their distorted stake.

Theorem 13. *Given protocol $\prod_{\mathsf{LM}}$ that UC-emulates $\mathtt{MAINTAIN\_LEDGER}$ in $\mathcal{G}_{\mathsf{PL}}$ with $\mathcal{F}_{\mathsf{stk}}$ and $\mathsf{Lkg} = \phi\left(\{S^i\}_{1\le i\le n}\right)$, replacing $\mathcal{F}_{\mathsf{stk}}$ in $\prod_{\mathsf{LM}}$ with $\mathcal{F}_{\mathsf{sd}}$ yields a protocol $\prod_{\mathsf{LM}^*}$ that UC-emulates $\mathtt{MAINTAIN\_LEDGER}$ with $\mathsf{Lkg} = \phi\left(\{\tilde{S}^i\}_{1\le i\le n}\right)$, where ϕ is a slot leader function mapping parties' stake distribution over time to a sequence of slot leaders, and S^i and $\tilde{S}^i$ denote P_i's true and distorted (by $\mathcal{F}_{\mathsf{sd}}$) stake history, respectively.*

We provide complete proof of Theorem 13 in Appendix C.3. As stake-related leakage occurs solely in leader election [KRDO17,KKKZ19,DGKR18], we consider the distorted stake to be used exclusively for the private lottery. For instance, to issue and validate payment transactions, one should keep using the true stake. This further suggests that other protocols of the standard design, i.e., the transaction submission and validation, undergo no alterations. In addition, by the composability of UC, if $\prod_{\mathsf{LM}^*}$ realizes $\mathcal{G}_{\mathsf{PL}}$ through calls to $\mathcal{F}_{\mathsf{sd}}$, and $\prod_{\mathsf{sd}}$ realizes $\mathcal{F}_{\mathsf{sd}}$, then substituting calls to $\mathcal{F}_{\mathsf{sd}}$ with subroutine calls to $\prod_{\mathsf{sd}}$ leads to a secure protocol for $\mathcal{G}_{\mathsf{PL}}$. This enables us to explicitly establish our design objectives as designing practical $\mathcal{F}_{\mathsf{sd}}$ that produces noisy stake satisfying Definition 10 and devising protocols that securely realize $\mathcal{F}_{\mathsf{sd}}$.

6.2 Differentially Private Stake Distortion

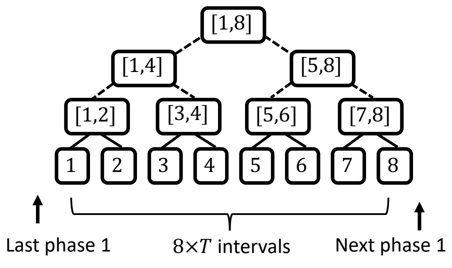
We start with two different designs of $\mathcal{F}_{\mathsf{sd}}$, namely, the timer ($\mathcal{F}_{\mathsf{Timer}}$) and the binary mechanism ($\mathcal{F}_{\mathsf{Bin}}$).

Timer mechanism $\mathcal{F}_{\mathsf{Timer}}$. In general, the mechanism periodically (every T time steps) distorts each party's stake with fresh Laplace noise drawn from $\mathsf{Lap}(\frac{\alpha}{\epsilon})$. For other times between two distortions, it reuses the noisy stake in the previous round. The rationale for distorting stake in a periodic manner (as opposed to distorting every time) is twofold. First, parties' stake generally exhibits stability over short intervals. Insignificant stake changes exert negligible effects on their lottery winning odds, rendering the release of a new distorted stake each time unwarranted. Second, distorting stake in a less frequent manner yields a smoother privacy decay (Section 6.3), otherwise stake privacy may drop significantly over a short time. Note that the distorted stake can be negative due to the symmetry of the Laplace noise. We stress that a party with a negative stake is treated the same as no stake, and thus the corresponding party has no chance of winning the following private lottery. For the formal $\mathcal{F}_{\mathsf{Timer}}$ formulation, please refer to Appendix B.2.

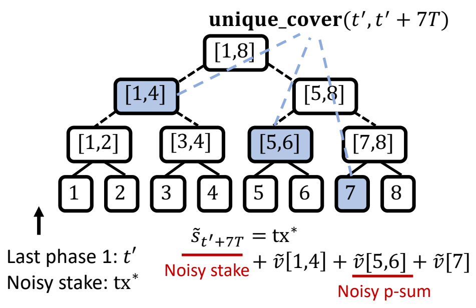
Theorem 14. *Protocol $\prod_{\mathsf{LM}^*}$ mounted with $\mathcal{F}_{\mathsf{Timer}}$ is (α, ϵ)-private with privacy decay $\omega(t-u) = \lfloor \frac{t-u}{T} \rfloor$.*

By the theory of the the Laplace mechanism [DR+14], the privacy loss for all stake values within the latest T time steps is bounded by ϵ. Moreover, due to multiple releases of the noisy stake at different times, the privacy loss of any historical stake follows the privacy loss under k-fold composition rules [DR+14]. Thus, one can obtain a linear decay function such that $\omega(t-u) = \lfloor \frac{t-u}{T} \rfloor$. We defer the complete proof of Theorem 14 in Appendix C.3.

Binary mechanism $\mathcal{F}_{\mathsf{Bin}}$. We improve upon $\mathcal{F}_{\mathsf{Timer}}$ and provide the binary (tree) mechanism, $\mathcal{F}_{\mathsf{Bin}}$, which provides a smoother (logarithmic) privacy decay over time. In general, the mechanism relies on the fact that any party's stake is the additive outcome of all her transactions. For each party P_i, the functionality internally groups all P_i's transactions based on transaction time, wherein the additive outcome of each group represents a *partial sum (p-sum)* of P_i's current stake. Subsequently, $\mathcal{F}_{\mathsf{Bin}}$ distorts each p-sum using Laplace noises, and for each time, the noisy stake is obtained by aggregating a set of noisy p-sums. Moreover, a single noisy p-sum may be reused to derive multiple noisy stake at different times, which further leads to less privacy loss over time.



(a) Binary interval tree


unique_cover($t', t'+7T$)
[1,8]
[1,4]
[5,8]
[1,2]
[3,4]
[5,6]
[7,8]
1
2
3
4
5
6
7
8
Last phase 1: t'
Noisy stake: tx*
$\tilde{s}_{t'+7T}$ = tx*
+ $\tilde{v}[1,4]$ + $\tilde{v}[5,6]$ + $\tilde{v}[7]$
Noisy stake
Noisy p-sum

(b) Determine noisy p-sums

Fig. 2: Example of Binary Mechanism with $L = 8T$

Specifically, $\mathcal{F}_{\mathsf{Bin}}$ features two distortion phases. In phase one, $\mathcal{F}_{\mathsf{Bin}}$ releases a new distorted stake every $L \gg T$ time steps acting as a single noisy p-sum, reused for the next L steps. Phase two occurs between phase one distortions, with a new stake released every T time steps. Precisely, $\mathcal{F}_{\mathsf{Bin}}$ employs a binary interval tree, with leaf nodes representing time intervals of length T and internal nodes integrating their children's time intervals. Figure 2a shows an example. For every T time steps, $\mathcal{F}_{\mathsf{Bin}}$ identifies a set of disjoint tree nodes that *uniquely cover* the interval from the last phase one distortion to the current time. More specifically given the last phase one distortion time t' and the current time j, we use the following algorithm to cover $[t, j]$: (i) Set index $a = t'$, identify the sub-interval (tree node) $[a, b] \subseteq [a, j]$ encompassing $T \cdot 2^k$ time steps, with $T \cdot 2^{k+1} > |j-a|$; (ii) Update $a = b+1$ and repeat step (i) until the entire interval is covered. For better illustration, we show an example in Figure 2b. Note that, by running the aforementioned strategy, at most $\log_2(\frac{L}{T})$ nodes are required to uniquely any intervals. Subsequently, $\mathcal{F}_{\mathsf{Bin}}$ groups transactions based on selected nodes, computes noisy p-sums, and aggregates them into a new noisy stake. For the formal $\mathcal{F}_{\mathsf{Bin}}$ formulation, please see Appendix B.2.

For any stake s_u with $u \leq t$ (current time), the total privacy loss of s_u under $\mathcal{F}_{\mathsf{Bin}}$ comes from both distortion phases. By Theorem 14, the phase one privacy loss composes linearly. In phase two, the transaction outcome at time u impacts at most $\log_2(\frac{\max(t-u,L)}{T})$ noisy p-sums, thus leading to a logarithmic privacy decay. Combining the two we conclude the overall privacy decay as follows.

Theorem 15. *Protocol $\prod_{\mathsf{LM}^*}$ mounted with $\mathcal{F}_{\mathsf{Timer}}$ is (α, ϵ)-private with privacy decay*

$$\omega(t-u) \leq \begin{cases} \lfloor \frac{t-u}{L} \rfloor + \log_2(\frac{L}{T}), & \textit{if } (t-u) > L \\ \log_2(\frac{t-u}{T}), & \textit{otherwise} \end{cases} \tag{3}$$

Proof. Please refer to Appendix C.3 for the complete proof.

6.3 Analysis of the Stake Distortion

Resilience to SIA attacks. We first analyze how SIA errors change when stake distortion is employed. Theorem 7 implies that in standard protocols where no stake distortion is present, the SIA error can be made arbitrarily small as long as the attacker can flip the stake-based coins unlimited times over a long period of time (this allows the attackers to set θ and τ arbitrarily small). When switching to distorted stake, SIA only finds the interval where the distorted stake is located. At this point, the errors are based on two factors, namely, the errors inherent in the attack, and the errors due to the injected DP noises.

Theorem 16. *Given protocol* $\prod_{\mathsf{LM}}$ *with SIA error* Err. *Extending it to* $\prod_{\mathsf{LM}^*}$ *with* $\mathcal{F}_{\mathsf{Timer}}$ *and* $\mathcal{F}_{\mathsf{Bin}}$ *yields SIA errors in* $\mathsf{Err} + O_p\left(\frac{\alpha}{\epsilon}\right)$ [1]*, and* $\mathsf{Err} + O_p\left(\frac{\alpha}{\epsilon}\sqrt{\log_2(\frac{L}{T})}\right)$*, respectively.*

Theorem 16 is the direct application of Chernoff bound [HR70] (complete proofs in Appendix C.3) to the injected Laplace noises. Note that the SIA error bounds now rely on both the attacker-selected parameters (θ and τ) and the privacy parameters (α and ϵ), indicating that one may not achieve arbitrary precision in stake inferences with an unlimited number of simulated coin flips.

Safety analysis. It is possible that the adversary's stake after distortion, say $\tilde{f}$, is larger than the pre-distortion value f. However, as long as there is a limited number of parties (existing protocols typically bound this by imposing a minimum staking amount [CM16,Cha21], say υ), the bounded variance property of Laplace noise [DR+14] enables the derivation of a slack, i.e. by Chernoff inequalities, that bounds the stake increment $\tilde{f} - f$ with a high probability. By setting the target as $\tilde{f} < \frac{1}{3}$, and with the derived slack, one can obtain another threshold ξ such that as long as $f < \xi$, then $\tilde{f} < \frac{1}{3}$ with high probability, i.e., the safety remains valid after stake distortion. We summarize the safety bounds in Theorem 17 and show in Section 7 that they are reasonable through a case study.

Theorem 17. *Assuming* $0 < \beta < 1$, $\Gamma = \frac{1.74\alpha}{\epsilon S}\sqrt{\frac{S}{\upsilon}\log\frac{1}{\beta}} < 1$*, and* $f = \frac{1-\gamma}{3}$ *with* $\Gamma \leq \gamma < 1$ *(or* $\gamma \geq \sqrt{\log_2(\frac{L}{T})}\times\Gamma$*), the adversary's stake after distortion,* $\tilde{f}$*, is less than* $\frac{1}{3}$ *with a probability of at least* $1-\beta$ *under* $\mathcal{F}_{\mathsf{Timer}}$ *(or* $\mathcal{F}_{\mathsf{Bin}}$*).*

Theorem 17 demonstrates that for a protocol, $\prod_{\mathsf{LM}}$, tolerating up to $\frac{1}{3}$ malicious stake, employing stake distortion with $\mathcal{F}_{\mathsf{Timer}}$ and $\mathcal{F}_{\mathsf{Bin}}$ results in a modified protocol, $\prod_{\mathsf{LM}^*}$, which tolerates up to $\frac{1}{3} - O_p\left(\frac{\alpha}{\epsilon S}\sqrt{\frac{S}{\upsilon}}\right)$, and $\frac{1}{3} - O_p\left(\frac{\alpha}{\epsilon S}\sqrt{\frac{S}{\upsilon}\log_2(\frac{L}{T})}\right)$ malicious controlled stake, respectively (complete proofs in Appendix C.3). Moreover, Theorem 17 highlights a safety-privacy trade-off, where weaker privacy guarantees (small ϵ or large α) result in smaller γ, allowing for greater malicious tolerance. This offers practitioners the flexibility to balance safety and privacy, opting to exchange some safety for enhanced privacy, or vice versa.

Liveness analysis. Typically, the liveness guarantee of PoS ledgers is directly tied with the honest majority assumption (same as the safety guarantee) if no changes are made to the networking layer. For instance, by [DGKR18, Theorem 9] and [CM16, Theorem 4] the liveness is ensured for PoS protocols as long as the honest parties control more than a certain fraction of the total stake (i.e., $> \frac{2}{3}$). Note that our stake distortion does not alter the network assumptions (i.e., a network of authenticated multicast channels with bounded delayed [GOT19]), and by Theorem 17, the honest majority assumption still holds after stake distortions. Therefore, without loss of generality, in what follows we focus on the safety analysis and assume that once safety is preserved, liveness is also retained.

Impact on individual party. The noise introduced to each party's stake has a distinct effect on their odds of winning the slot leader election. Given the uniform variance of injected noise, smaller stakeholders might face evident stake distortion or changes in their winning odds for a single election round. Conversely, for larger stakeholders, the change in their winning odds might not be as prominent. Nevertheless, as the added noises possess zero means, for long-term execution, the expected number of election wins aligns with each party's true stake.

6.4 Stake Distortion Protocol Design

Building blocks. We build the protocols in a hybrid world where the following auxiliary functionalities are available: (i) There exists a hiding commitment scheme [SCG+14] that given randomness r and message x, an algorithm $\mathsf{Com}_r(x)$ commits x to com_x with opening r; (ii) We utilize pseudorandom functions [KKKZ19], $\mathsf{PRF}_{\mathsf{k}}(x)$ with input x and the evaluation key k, for deriving DP noises and rely on an unpredictable random oracle, $\mathcal{F}_{\mathsf{ro}}$, to produce randomness. An instantiation of $\mathcal{F}_{\mathsf{ro}}$ is the random beacon [BKNS22]; (iii) There exists a non-interactive zero-knowledge functionality [KKKZ19], $\mathcal{F}^{\mathcal{L}}_{\mathsf{nizk}}$, that allows proving of statements in an NP language

[1] O_p is the Big O in its probability notion [DC03]

$\mathcal{L}$; (iv) There is a "determine stake" functionality, $\mathcal{F}_{\mathsf{stk}}$, allows parties to access their stake, committed stakes, and stake commitment opening at any time. Additionally, $\mathcal{F}_{\mathsf{stk}}$ allows for querying a list, L, containing: (a) all registered users with their respective committed stake, and (b) each party's noise generation key. In practice, the aforementioned information can be efficiently computed from the blockchain [GOT19].

Timer protocol $\prod_{\mathsf{timer}}$. We provide the design of timer protocol $\prod_{\mathsf{timer}}$ that securely realizes $\mathcal{F}_{\mathsf{Timer}}$. Initially, we consider all parties agree on a global clock and each party derives a pair of noise generation keys $(\mathsf{n_{pk}}, \mathsf{n_{sk}})$ upon joining the system. Such key pair is sampled by selecting a random private key $\mathsf{n_{sk}}$ and setting $\mathsf{n_{pk}} \leftarrow \mathsf{PRF}_{\mathsf{n_{sk}}}(0)$. The parties manage $\mathsf{n_{sk}}$ themselves, which remains unknown to others, but $\mathsf{n_{pk}}$ is accessible to other parties (i.e., through $\mathcal{F}_{\mathsf{stk}}$). Parties can join at any time, while stake distortion occurs every T steps, thus we restrict each new party to wait until the next distortion schedule to acquire the first distorted stake before it can participate in the ledger maintenance. Protocol 6.2 shows details from the viewpoint of party P with identifier pid.

Protocol 6.2: Timer protocol $\prod_{\mathsf{Timer}}$

```
    Upon receiving DISTORT from pid
1:  Obtain current time $j$ from the global clock
2:  if $j \mod T == 0$ then $(s_j, \mathsf{com}_{s_j}, r) \leftarrow \mathcal{F}_{\mathsf{stk}}$
3:      Generate private randomness: $(\mathsf{ra}, \mathsf{rb}) \leftarrow \mathcal{F}_{\mathsf{ro}}(j)$, $(z^0, z^1) \leftarrow \mathsf{PRF}_{\mathsf{n_{sk}}}(\mathsf{ra}, \mathsf{rb})$
4:      Transform private randomness to DP noises: $\tilde{s}_j \leftarrow s_j + \frac{\alpha}{\epsilon}(\ln(z^0) - \ln(z^1))$
5:      Sample $\tilde{r}$ and $\mathsf{com}_{\tilde{s}_j} \leftarrow \mathsf{Com}_{\tilde{r}}(\tilde{s}_j)$
6:      Let $\mathbf{x} = (\epsilon, \alpha, \mathsf{n_{pk}}, \mathsf{com}_{s_j}, \mathsf{com}_{\tilde{s}_j}, \mathsf{ra}, \mathsf{rb})$, and $\mathbf{w} = (s_j, \tilde{s}_j, \mathsf{n_{sk}}, z^0, z^1, r, \tilde{r})$
7:      Generate zero-knowledge proof: $\pi_j \leftarrow \mathcal{F}_{\mathsf{nizk}}^{\mathcal{L}_{\mathsf{timer}}}(\mathsf{prove}, \mathbf{x}, \mathbf{w})$
8:      return $\tilde{s}_j$ broadcast $(\mathsf{pid}, \mathsf{com}_{\tilde{s}_j}, \pi_j)$
9:  else $\tilde{s}_j = \tilde{s}_{j-1}$ return $\tilde{s}_j$

    Upon receiving (GET_COMM, pid)
10: Retrieve $(\mathsf{com}_{\tilde{s}_j}, \pi_j)$ for pid from the network.
11: if accept $\leftarrow \mathcal{F}_{\mathsf{nizk}}^{\mathcal{L}_{\mathsf{timer}}}(\mathsf{verify}, \pi_j)$ then
12:     Record $\mathsf{com}_{\tilde{s}_j}$ for pid
```

In order to distort the stake, $\prod_{\mathsf{Timer}}$ first acquires necessary information from $\mathcal{F}_{\mathsf{stk}}$ (or from the blockchain) and obtains the current public randomness from $\mathcal{F}_{\mathsf{ro}}$. The protocol then evaluates a PRF to derive private random seeds[2], z^0, z^1, which are subsequently converted into an instance of Laplace noise drawn from $\mathsf{Lap}(\frac{\alpha}{\epsilon})$ [Ros14]. Given that $\mathsf{n_{sk}}$ remains concealed from other parties, no entity can gain knowledge about the derived DP noise. Furthermore, provided the output of $\mathcal{F}_{\mathsf{ro}}$ is unpredictable, the adversary is unable to adaptively select $\mathsf{n_{sk}}$ to maximize the derived DP noise. Lastly, $\prod_{\mathsf{Timer}}$ returns the distorted stake to party P and broadcasts a hiding commitment to the distorted stake, accompanied by a zero-knowledge proof demonstrating that the committed stake has been accurately distorted to all other parties. Next, we define the statements by their corresponding NP languages: A tuple $(\mathbf{x}, \mathbf{w}) \in \mathcal{L}_{\mathsf{timer}}$ if all the following holds:

– Instance: $\mathbf{x} = (\epsilon, \alpha, \mathsf{n_{pk}}, \mathsf{com}_{s_j}, \mathsf{com}_{\tilde{s}_j}, \mathsf{ra}, \mathsf{rb})$
– Witness: $\mathbf{w} = (s_j, \tilde{s}_j, \mathsf{n_{sk}}, z^0, z^1, r, \tilde{r})$
– Correctness of the noise generation key: $\mathsf{PRF}_{\mathsf{n_{sk}}}(0) = \mathsf{n_{pk}}$
– Correctness of the randomness for deriving DP noises: $\mathsf{PRF}_{\mathsf{n_{sk}}}(\mathsf{ra}, \mathsf{rb}) = (z^0, z^1)$
– Correctness of the stake distortion: $\tilde{s}_j = s_j + \frac{\alpha}{\epsilon}(\ln(z^0) - \ln(z^1))$
– $\exists\, r, \tilde{r}$ s.t. $\mathsf{com}_{s_j} = \mathsf{Com}_r(s_j)$ and $\mathsf{com}_{\tilde{s}_j} = \mathsf{Com}_{\tilde{r}}(\tilde{s}_j)$

For any other times between two distortion schedules, the protocol simply reuses the distorted stake from the last distortion round. Furthermore, when a party receives a broadcast stake commitment from others, the protocol interacts with $\mathcal{F}_{\mathsf{nizk}}^{\mathcal{L}}$ to verify the validity of π, accepting the commitment only if π is deemed valid.

Binary protocol $\prod_{\mathsf{Bin}}$. To implement $\prod_{\mathsf{Bin}}$ that securely realizes $\mathcal{F}_{\mathsf{Bin}}$, we utilize the same strategy as $\prod_{\mathsf{Timer}}$ to obtain randomness and transform DP noises. The tricky part is dealing with the DP interval tree and deriving noisy p-sums. In our design, we use the binary representation of time to implicitly track the tree structure [CSS+10], which simplifies the NP statements for generating proofs. The details are provided in Protocol 6.3.

[2] We treat random seeds as fixed-point values in $(0, 1)$.

Protocol 6.3: Binary protocol $\prod_{\mathsf{Bin}}$

```
    //vectors tx, and $\tilde{\mathsf{tx}}$ are initiated as $\overrightarrow{0}$.
    Upon receiving DISTORT from pid
1:  Obtain current time $j$ from the global clock.
2:  if $j \mod L = 0$ then
3:      Follow $\prod_{\mathsf{Timer}}$ steps 2-10, and cache $\tilde{s}_j$ as $\mathsf{tx}^*$
4:  if $t \bmod T = 0$, where $t \leftarrow j \bmod L$ then
5:      $(s_j, \mathsf{com}_{s_j}, r), (s_{j-T}, \mathsf{com}_{s_{j-T}}, r') \leftarrow \mathcal{F}_{\mathsf{stk}}$
6:      Sample private randomness: $(\mathsf{ra}, \mathsf{rb}) \leftarrow \mathcal{F}_{\mathsf{ro}}(j)$, $(z^0, z^1) \leftarrow \mathsf{PRF}_{\mathsf{n_{sk}}}(\mathsf{ra}, \mathsf{rb})$
7:      Express current time $t$ in binary form: $t = \sum_k 2^k \cdot \mathsf{bin}_k(t)$
8:      Set $\ell$ as the first non-zero bit: $\ell \leftarrow \min\{k : \mathsf{bin}_k(t) \neq 0\}$
9:      $\mathsf{tx}[\ell] \leftarrow \sum_{k<\ell} \mathsf{tx}[k] + (s_j - s_{j-T})$
10:     for $k < \ell$ override $\tilde{\mathsf{tx}}[k] = \mathsf{tx}[k] = 0$; $\tilde{\mathsf{tx}}[\ell] = \mathsf{tx}[\ell] + \frac{\alpha}{\epsilon}(\ln(z^0) - \ln(z^1))$
11:     Release the noisy stake from noisy p-sums: $\tilde{s}_j \leftarrow \mathsf{tx}^* + \sum_{k:\mathsf{bin}_k(t)=1} \tilde{\mathsf{tx}}[k]$;
12:     Sample commitment randomness $\tilde{r}, r_\ell, \tilde{r}_\ell$
13:     $\mathsf{com}_{\tilde{s}_j} \leftarrow \mathsf{Com}_{\tilde{r}}(\tilde{s}_j)$, $\mathsf{com}_{\mathsf{tx}_k} \leftarrow \mathsf{Com}_{r_\ell}(\mathsf{tx}[\ell])$, and $\mathsf{com}_{\tilde{\mathsf{tx}}_\ell} \leftarrow \mathsf{Com}_{\tilde{r}_\ell}(\tilde{\mathsf{tx}}[\ell])$.
14:     Let $\mathbf{x} = (\epsilon, \alpha, \mathsf{n_{pk}}, \mathsf{ra}, \mathsf{rb}, \mathsf{com}_{s_j}, \mathsf{com}_{s_{j-T}}, \mathsf{com}_{\tilde{s}_j}, \mathsf{com}_{\mathsf{tx}}, \mathsf{com}_{\tilde{\mathsf{tx}}})$
15:     Let $\mathbf{w} = (s_j, s_{j-T}, \tilde{s}_j, \mathsf{tx}, \tilde{\mathsf{tx}}, \mathsf{n_{sk}}, z^0, z^1, r, \tilde{r}, \overrightarrow{r})$
16:     Get zero-knowledge proof: $\pi_j \leftarrow \mathcal{F}_{\mathsf{nizk}}^{\mathcal{L}_{\mathsf{bin}}}(\mathsf{prove}, \mathbf{x}, \mathbf{w})$
17:     return $\tilde{s}_j$
18:     broadcast $(\mathsf{pid}, \mathsf{com}_{\tilde{s}_j}, \mathsf{com}_{\mathsf{tx}_\ell}, \mathsf{com}_{\tilde{\mathsf{tx}}_\ell}, \pi_j)$
19: else $\tilde{s}_j = \tilde{s}_{j-1}$ return $\tilde{s}_j$

    Upon receiving (GET_COMM, pid)
20: retrieve commitments $\mathsf{com}_{\tilde{s}_j}, \mathsf{com}_{\mathsf{tx}_\ell}, \mathsf{com}_{\tilde{\mathsf{tx}}_\ell}$ , and the proof $\pi_j$ for pid from the network
21: override the cached commitments $\forall_{k<\ell}$ $\mathsf{com}_{\mathsf{tx}_k} = \mathsf{com}_{\tilde{\mathsf{tx}}_k} = \mathsf{Com}(0)$ for pid
22: if accept $\leftarrow \mathcal{F}_{\mathsf{nizk}}^{\mathcal{L}_{\mathsf{bin}}}(\mathsf{verify}, \pi_j)$ then
23:     Record $\mathsf{com}_{\tilde{s}_j}, \mathsf{com}_{\mathsf{tx}_{0:\ell}}$, and $\mathsf{com}_{\tilde{\mathsf{tx}}_{0:\ell}}$ for pid
```

Phase one distortion in our design follows the same implementation as $\prod_{\mathsf{Timer}}$, except the noisy stake is cached in tx^*. In phase two, the protocol derives a new noisy p-sum every T time steps based on the current time's binary representation (6.3:5-11). $\prod_{\mathsf{Bin}}$ then aggregates a subset of previously computed (cached) noisy p-sums to determine the noisy stake (6.3:12). Note that, $\prod_{\mathsf{Timer}}$ can safely recycle a subset of "old" p-sums (6.3:10), requiring it to track at most $\log(\frac{L}{T})$ noisy p-sums. Similarly, $\prod_{\mathsf{Bin}}$ generates proof for validating the stake distortion process, with NP statements defined as follows: A tuple $(\mathbf{x}, \mathbf{w}) \in \mathcal{L}_{\mathsf{bin}}$ if all the following holds:

- $\mathbf{x} = (\epsilon, \alpha, \mathsf{n_{pk}}, \mathsf{ra}, \mathsf{rb}, \mathsf{com}_{s_j}, \mathsf{com}_{s_{j-T}}, \mathsf{com}_{\tilde{s}_j}, \mathsf{com}_{\mathsf{tx}}, \mathsf{com}_{\tilde{\mathsf{tx}}})$
- $\mathbf{w} = (s_j, s_{j-T}, \tilde{s}_j, \mathsf{tx}, \tilde{\mathsf{tx}}, \mathsf{n_{sk}}, z^0, z^1, r, \tilde{r}, \overrightarrow{r})$
- $\mathsf{PRF}_{\mathsf{n_{sk}}}(0) = \mathsf{n_{pk}}$ and $\mathsf{PRF}_{\mathsf{n_{sk}}}(\mathsf{ra}, \mathsf{rb}) = (z^0, z^1)$
- Correct new p-sum: $tx[\ell] = \sum_{k=0}^{\ell} \mathsf{tx}[k] + s_j - s_{j-T}$
- Noisy p-sum: $\tilde{tx}[\ell] \leftarrow \mathsf{tx}[\ell] + \frac{\alpha}{\epsilon}(\ln(z^0) - \ln(z^1))$
- Correct noisy stake: $\tilde{s}_j \leftarrow \mathsf{tx}^* + \sum_{k:\mathsf{bin}_k(t)=1} \tilde{\mathsf{tx}}[k]$
- $\exists$ opening $r, \tilde{r}$ s.t. opens $\mathsf{com}_{s_j}, \mathsf{com}_{\tilde{s}_j}$ to $s_j, \tilde{s}_j$.
- $\forall_k$ ($\exists$ $r_k, \tilde{r}_k$ s.t. opens $\mathsf{com}_{\mathsf{tx}_k}, \mathsf{com}_{\tilde{\mathsf{tx}}_k}$ to $\mathsf{tx}[k], \tilde{\mathsf{tx}}[k]$)

Observe that $\prod_{\mathsf{Bin}}$ requires verifying nodes to cache additional objects per party, i.e., the commitments to tx, $\tilde{tx}$. However, one needs to keep track of at most $\log(\frac{L}{T})$ p-sums(commitments). Hence, the storage blowup in contrast to $\prod_{\mathsf{Timer}}$ is bounded by $O(\log(\frac{L}{T}))$.

Theorem 18. *$\prod_{\mathsf{Timer}}$ and $\prod_{\mathsf{Bin}}$ UC emulates [Can01] $\mathcal{F}_{\mathsf{Timer}}$, and $\mathcal{F}_{\mathsf{Bin}}$, respectively, in the $(\mathcal{F}_{\mathsf{stk}}, \mathcal{F}_{\mathsf{ro}}, \mathcal{F}_{\mathsf{nizk}}^{\mathcal{L}})$-hybrid world with the presence of a p.p.t. adversary.*

Proof. Please refer to the Appendix B.3

7 Case Study with Real-World System

In this section, we perform case studies on our stake distortion protocols using a real-world PoS blockchain, Ethereum 2.0 [Fou22], and examine its impact on original guarantees. We explore the practicality of stake distortion by addressing several key questions.

- Q1. Will the ledger still provide safety guarantees after adopting stake distortion? What privacy level can be achieved given certain safety requirements?
- Q2. How does $\mathcal{F}_{\text{Timer}}$ compare to $\mathcal{F}_{\text{Bin}}$ in privacy decay? How one can benefit from a smoother decay?
- Q3. Does stake distortion mitigate the SIA risks?

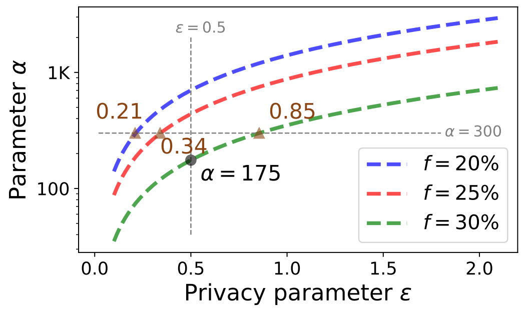
Setup. We implement simulations based on Ethereum 2.0's phase 0 specification [Cha21] and utilize key statistics from the beacon chain [Eth22]. With an average daily transaction volume per unique address below 0.25, it's reasonable to assume that Ethereum 2.0 parties' stakes remain stable over a 4-day period. Consequently, we set the distortion schedule $T = 4$ days and $\mathcal{F}_{\text{Bin}}$'s phase 1 interval $L = 45T$ (6 months). Table 1 summarizes the setup.

Total stake, S	13,488,174 Eth
Minimum stake, υ	32 Eth
Slot time (delay)	12 seconds
Distortion interval, T	4 days
$\mathcal{F}_{\text{Bin}}$ Phase 1 interval, L	$45 \times T$

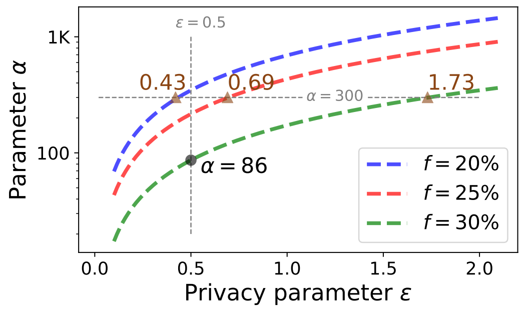
Table 1: Ethereum 2.0 parameters setup

7.1 Safety Guarantee

We plot safety curves in Figure 4 based on Theorem 17 with $\delta = 10^{-9}$ (safety failure probability),[3] and assume $\tilde{f} < \frac{1}{3}$ as the general safety requirement. Given α (resp. ϵ), the additive bound for distinguishing two stake values, the curves imply the minimum privacy loss, ϵ (resp. the maximum α), achievable at certain safety levels. For each mechanism, we plot three curves, which correspond 20%, 25%, and 30% of malicious tolerance, respectively.



(a) Timer, $\mathcal{F}_{\text{Timer}}$



(b) Binary, $\mathcal{F}_{\text{Bin}}$

Fig. 3: Safety upper bounds

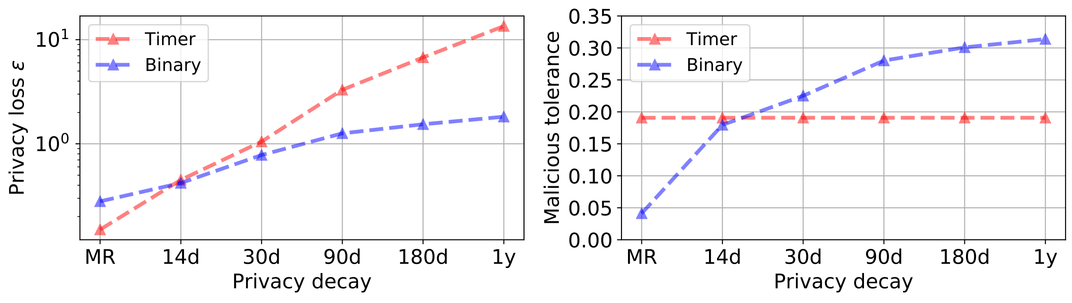
Observation-1. Both mechanisms provide reasonable stake privacy while retaining safety at the same time. In general, even with a relatively small ϵ and a large malicious threshold, i.e., $\epsilon = 0.5$ [4], and $f = 30\%$, α can be set up to 175 and 86, respectively, for $\mathcal{F}_{\text{Timer}}$ and $\mathcal{F}_{\text{Bin}}$. By the latest blockchain data [DMNP21], only less than 0.05% of accounts have more than 80 Eth,which suggests that privacy is ensured for at least 99.95% of total users.

Observation-2. There is a trade-off between privacy and safety guarantees. Apparently, one can achieve a larger α or a smaller privacy loss when considering a relatively weaker safety (smaller malicious tolerance). This suggests a trade-off between privacy and safety which provide flexibility for practitioners to better configure the protocols. For example, if there is a slashing mechanism [AKS21] to help reduce the malicious fraction, one may choose to gain better privacy by trading off some safety.

Given that the two mechanisms have different privacy decay, to better study them, we compare $\mathcal{F}_{\text{Timer}}$ and $\mathcal{F}_{\text{Bin}}$ under different privacy decay goals, i.e., privacy loss for the most recent (MR) time, the last 14, 30, 90, 180 and 365 days. First, we bound the malicious tolerance, i.e. 20%, and set $\alpha = 200$ then plot their privacy loss over time in Figure 4a. Next, we compare them in a different angle in Figure 4b, where we keep the configurations of $\mathcal{F}_{\text{Timer}}$ unchanged but modify $\mathcal{F}_{\text{Bin}}$ to enforce it reaches the same privacy loss as $\mathcal{F}_{\text{Timer}}$ under the different privacy decay goals. We then compare the malicious tolerance (safety) ensured by two mechanisms.

[3] Similar to a practical bound chosen by Algorand [CM16]

[4] In comparison with Google community mobility report which uses $\epsilon = 2.4$ per user per day, $\epsilon = 0.5$ is quite small.



(a) Privacy loss vs. Time **(b)** Safety vs. privacy decay

Fig. 4: Timer vs. Binary mechanism

Observation-3. For short-term privacy, $\mathcal{F}_{\mathsf{Timer}}$ outperforms $\mathcal{F}_{\mathsf{Bin}}$, while the result is reversed when focusing on long-term privacy. As shown in Figure 4a, $\mathcal{F}_{\mathsf{Timer}}$ exhibits lower short-term privacy loss than $\mathcal{F}_{\mathsf{Bin}}$ when fixing the safety level. This occurs since $\mathcal{F}_{\mathsf{Timer}}$ adds a single instance of DP noise, leading to smaller adversary stake growth and enabling the configuration under a smaller ϵ. However, due to logarithmic privacy decay, $\mathcal{F}_{\mathsf{Bin}}$ offers better long-term privacy loss even with a larger ϵ. Figure 4b shows that if one is more concerned about long-term privacy, it is better to employ $\mathcal{F}_{\mathsf{Bin}}$ as it ensures a larger malicious tolerance, which up to 1.63× that of $\mathcal{F}_{\mathsf{Timer}}$. Overall, this observation provides practitioners with a sense of how they would benefit from a stake distortion that has a smoother privacy decay.

7.2 Safety Simulation

To further validate the safety bounds we conduct stake distortion simulations. Specifically, we simulate a set of stakeholders and assume the attacker corrupts different fractions of them. Subsequently, we configure ϵ, α as per Theorem 17 and run stake distortion over each party. Finally, we investigate whether the corrupted stake after distortion, $\tilde{f}$, exceeds 1/3. Additionally, to max out the attacker's stake increment, we consider each individual stakeholder to hold a minimum staking, and thus the number of parties is maxed out as $n = \frac{S}{v}$. Table 2 summarizes the result over 10K repeated runs.

	Timer $\mathcal{F}_{\mathsf{Timer}}$		Binary $\mathcal{F}_{\mathsf{Bin}}$	
f (%)	(ϵ, α)	max $\tilde{f}$	(ϵ, α)	max $\tilde{f}$
10%	(0.5, 1214)	0.33082	(0.5, 552)	0.33078
15%	(0.5, 963)	0.32952	(0.5, 475)	0.32869
20%	(0.5, 701)	0.33055	(0.5, 346)	0.33132
25%	(0.5, 438)	0.32858	(0.5, 216)	0.32336
30%	(0.5, 175)	0.32309	(0.5, 86)	0.32626

Table 2: Simulation of stake distortion

Observation-4. With proper setups as per the safety bounds, $\hat{f} < \frac{1}{3}$ with overwhelming probability. According to Table 2, the corrupted stake, $\tilde{f}$, after distortion for all testing groups does not exceed 0.3314, which is consistent with the primary safety requirement, i.e., $\tilde{f} < 1/3$. Consider that stake distortions occur every 4 days, thus the 10K repeated runs can actually simulate a 109-year operation of the blockchain coupled with stake distortion. The simulation implies in those 109 years of operation, there has not been even one violation of the safety requirement. In fact, as β is set to 10^{-9}, the probability of even one violation happens should smaller than 10^{-4} (taking union bound). In general, we conclude that the simulations validate our safety upper bounds.

7.3 Resilience to SIA

We investigate whether stake distortion helps mitigate SIA risk, which is achieved by initiating RdBin on simulated Ethereum 2.0's ledger maintenance protocol with and without stake distortion, then compare their relative inference errors. Moreover, we are also interested in the SIA resistance of $\mathcal{F}_{\mathsf{Timer}}$ and $\mathcal{F}_{\mathsf{Bin}}$ under different safety requirements. To address these, we simulate the protocol for 1.29×10^6 steps (6 months in the real world) and issue one RdBin attack every 4 days. By default, we configure RdBin with $\theta = v$, $\tau = 0.01$, and run the attack against a target with random stake $f_v \in (0, f)$. When simulating Ethereum 2.0 with stake distortion, we assume the protocol tolerates different malicious fractions, i.e., from 10% to 30%, configure related parameters accordingly, and run attacks against each group independently. We report the relative inference errors of RdBin in Figure 5.

Observation-5. Both $\mathcal{F}_{\mathsf{Timer}}$ and $\mathcal{F}_{\mathsf{Bin}}$ mitigate SIA risk, and the simulation result suggests a trade-off between safety guarantees and SIA resistance. According to Figure 5, when stake distortion is employed, the relative

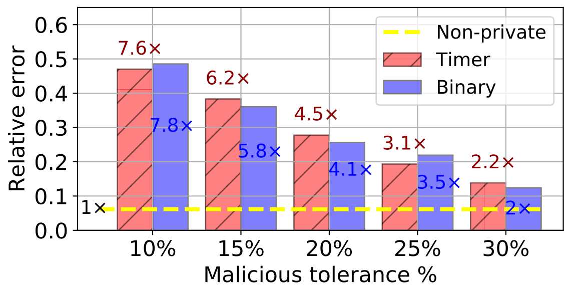


Fig. 5: Privacy loss vs. Time

SIA inference error is at least twice that of the standard protocol and can reach up to 7.8×. This further implies the practicability of stake distortion in mitigating SIA risks. Moreover, the relative error of both $\mathcal{F}_{\mathsf{Timer}}$ and $\mathcal{F}_{\mathsf{Bin}}$ decreases when the malicious tolerance increases, i.e., the weaker the safety demanded, the stronger the resistance to SIA obtained, and vice versa.

7.4 Key Takeaways

The first two observations address Q1, where we learn that our mechanisms provide reasonable privacy while preserving safety (liveness). Moreover, we are aware of a privacy-safety tradeoff in our mechanisms. Observation 3 compares the two mechanisms and addresses Q2, from which we learn that one should choose $\mathcal{F}_{\mathsf{Timer}}$ when focusing on short-term privacy goals, while $\mathcal{F}_{\mathsf{Bin}}$ is a better option for long-term goals. The safety simulations (Section 7.2) validate our theoretical bounds while the SIA experiments (Section 7.3) address Q3 and demonstrate our mechanisms indeed mitigate SIA risks.

8 Performance Evaluation

Implementation and configuration. We utilize the same method as [SCG⁺14] to implement PRF and Com using SHA256 compression functions. Fractional numbers are stored as 32-bit fixed-point values with a scaling factor of 2^{16}, and we consider the public randomness as 16-bit unsigned fixed-point values with a scaling factor of 2^{16}, ensuring the randomness is spread across $(0, 1)$. We use the Remez algorithm [Taw05] to create approximated polynomials for the $\ln(x)$ circuit. Alternative techniques [NFPH15,Shi11] and engineering optimizations [fas21] exist for implementing the $\ln(x)$ circuit, potentially improving approximation accuracy or computational efficiency. Developing optimized protocols is not the primary focus of this work, but may be of independent interest. Nevertheless, even without these optimizations, our evaluations still show a reasonable overhead associated with the stake distortion protocols. To implement the NIZK proof, we utilize zk-SNARK library Zokrates-0.8.5 [zok22] under the proving scheme Groth16 [Gro16]. In addition, we assume the existence of the parties' stake commitments and their integrity have been verified, as in practice such commitments can be effectively obtained and verified [SCG⁺14] from the blockchain. The protocols are evaluated over machines with 2.6GHz CPU and 16Gb RAM.

		Timer	**Binary**
Setup	**Setup time**	19.64 s	20.63 s
	Proving key size	38 Mb	38 Mb
	Verification key size	4 Kb	4 Kb
Prove	**Proof time**	7.07 s	7.32 s
	Proof size	4 Kb	4 Kb
Verify	**Verification time**	11 ms	10ms
Other	**Time**	1.33 s	1.89 s
	Storage	22.72 Mb	62.21 Mb

Table 3: Evaluation result for the prototype protocol

Evaluation result. Table 3 summarizes the evaluation results, apparently, the performance overhead is dominated by the costs associated with the NIZK proof. Nevertheless, the adoption of our protocol may not impose an excessive performance overhead, for example, Zcash requires more than 120 seconds of proof time (without optimization) per block [SCG⁺14]. Thus, adopting our protocols only increase the proof overhead by at most 5.8%. Furthermore, since stake distortion doesn't transpire at every time step, the amortized overhead may be reduced. For example, assuming stake distortion occurs every 4 days, the amortized overhead in the Zcash case could be less than 0.002%. Note that stake distortion proofs can be combined with other ledger-specified proofs, enabling the verifier to validate all SNARK arguments via a single verification. For the storage overhead, both protocols require parties to cache additional data objects, i.e., noisy stake, commitments, etc. However, the storage volume does not exceed 63 Mb, which represents a slight overhead compared to storing the full blockchain data (i.e., 895 Gb for Ethereum [Eth22]).

9 Discussion

Mitigating privacy decay. One can create a stake distortion strategy without cumulative privacy loss over time. For example, for every T time steps, the party generates a noisy p-sum representing all transaction outcomes from the last T steps, and the noisy stake is derived by summing up all released noisy p-sums. This approach guarantees that distortion takes place across distinct transactions, and the extraction of the noisy stake is simply a post-processing step involving noisy p-sums. As per parallel composition and post-processing theorems [DR+14] of DP, the total privacy loss is bounded by ϵ. Although privacy loss doesn't accumulate, injected noise variance does, leading to reduced safety guarantees over time, i.e., the upper bound of adversarial stake increment increases. Another alternative is adopting the relaxed (ϵ, δ)-DP guarantee, allowing unbounded privacy loss with a small failure probability δ. This allows injecting Gaussian noises (offer tighter composition bounds than Laplace noises) that optimize privacy decay (see Appendix E for more details).

Permissionless clock synchronization. This work considers the globally synchronized setting in line with SOTA designs [KRDO17,DGKR18,KKKZ19] such that all parties can retrieve time from a global clock (functionality). We say that one can also adopt permissionless clock synchronization protocols [BGK+19], i.e. if some (honest) party believes the global time as j, all parties believe it as $j \pm \delta_c$, with a small δ_c, if a global clock is not accessible. Although, there might be small chances for a party that fails the slot leader election due to the use of outdated noisy stake, i.e. within δ_c right after a (globally defined) distortion schedule. Since δ_c is typically small [BGK+19] and the stake distortion interval T is relatively large, thus for the majority of time slots, all parties agree on the most recent distorted stake. While integrating our design with permissionless clock synchronization is not this paper's focus, it is considered an important future enhancement.

10 Related Works

Proof-of-Stake private ledgers. The first PoS design appears in [KN12]. Followed by this, there has been a series of efforts [BLMR14,CM16,BPS16,KRDO17,DGKR18,BGK+18,GKZ19,Cha21,GN17,Zam17] on formulating PoS models as well as providing protocols with provable security guarantees. Recently, several works have been proposed to address the stake and transaction privacy of PoS ledgers, which include Ouroborous Crypsinous [KKKZ19], Ganesh et al. [GOT19], and Baldimtsi et al. [BMSZ20]. However, the stake privacy is typically ensured at the cost of assuming complete anonymity of parties' identity [GOT19,KKKZ19]. Unfortunately, [KMNS21] demonstrates the existence of a tension between liveness and anonymity. In this work, we provide the first-of-its-kind solution that provides provable stake privacy for PoS private ledgers that do not assume anonymity.

Stake inference attacks. Both [KKKZ19] and [GOT19] state that the parties' stake information may be revealed with the execution of the PoS protocols, no specific attacks for inferring stake have been proposed until the publication of [KMNS21]. However, their proposed attack considers only deterministic protocols and requires linear time complexity. In this work, we provide a practical stake inference attack that is proven to be valid against randomized protocols with probabilistic liveness. Moreover, our attack only yields sublinear time complexity.

Differential privacy. Differential privacy (DP) introduced by Dwork et.al. [DR+14] is currently the "de-facto" standard for achieving data privacy. Since 2010, a couple of efforts have been proposed to address DP under dynamic setting (or under the continual observation) [DNPR10,Dwo10,CSS+10,BFM+13,KPXP14,WBNM21,WBNM22]. Several works [BFM+13,KPXP14,LSV+19b] employs a relaxed privacy with decay model to ensure better utility, which is similar to our formulation. Nevertheless, as far as we know, we are the first to incorporate DP with private PoS blockchains.

11 Conclusion

In this paper, we present the first practical stake inference attack against randomized PoS protocols. The existence of such an attack further implies the inadequacy of SOTA PoS designs in achieving stake and transaction privacy. To formulate a rigorous privacy definition, we incorporate DP with the standard UC definition of private ledgers. Guided by the derived privacy model, we design DP stake distortion protocols that assist existing PoS protocols in resolving stake and transaction privacy.

Acknowledgment

This proposal was funded in part by NSF Awards 2016393 and 2237814.

References

AKS21. J.-P. Aumasson, D. Kolegov, and E. Stathopoulou. Security review of ethereum beacon clients. *arXiv preprint arXiv:2109.11677*, 2021.

BCG+19. S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, and H. Wu. Z: Enabling decentralized private computation. In *2020 IEEE Symposium on Security and Privacy*, pages 947–964, 2019.

Bel20. M. Bellare. Lectures on nizks: A concrete security treatment. 2020.

BFM+13. J. Bolot, N. Fawaz, S. Muthukrishnan, A. Nikolov, and N. Taft. Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory*, pages 284–295, 2013.

BGK+18. C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 913–930, 2018.

BGK+19. C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas. Ouroboros chronos: Permissionless clock synchronization via proof-of-stake. *Cryptology ePrint Archive*, 2019.

BGM16. I. Bentov, A. Gabizon, and A. Mizrahi. Cryptocurrencies without proof of work. In *International conference on financial cryptography and data security*, pages 142–157. Springer, 2016.

BKNS22. A. Bhat, A. Kate, K. Nayak, and N. Shrestha. Optrand: Optimistically responsive distributed random beacons. *Cryptology ePrint Archive*, 2022.

BLMR14. I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. *ACM SIGMETRICS Performance Evaluation Review*, 42(3):34–37, 2014.

BMSZ20. F. Baldimtsi, V. Madathil, A. Scafuro, and L. Zhou. Anonymous lottery in the proof-of-stake setting. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, pages 318–333. IEEE, 2020.

BMTZ17. C. Badertscher, U. Maurer, D. Tschudi, and V. Zikas. Bitcoin as a transaction ledger: A composable treatment. In *Annual international cryptology conference*, pages 324–356. Springer, 2017.

Box58. G. E. Box. A note on the generation of random normal deviates. *Ann. Math. Statist.*, 29:610–611, 1958.

BPS16. I. Bentov, R. Pass, and E. Shi. Snow white: Provably secure proofs of stake. *IACR Cryptol. ePrint Arch.*, 2016(919), 2016.

Buc16. E. Buchman. *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis, University of Guelph, 2016.

Can01. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

CDE+16. K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer, et al. On scaling decentralized blockchains. In *International conference on financial cryptography and data security*, pages 106–125. Springer, 2016.

Cha21. E. . P. . B. Chain. get_total_balance. `https://github.com/ethereum/consensus-specs/blob/dev/specs/phase0/beacon-chain.md`, 2021.

CM16. J. Chen and S. Micali. Algorand. *arXiv preprint arXiv:1607.01341*, 2016.

CSS+10. H. Chan, E. Shi, D. Song, et al. Private and continual release of statistics. In *International Colloquium on Automata, Languages, and Programming*, pages 405–417. Springer, 2010.

DC03. Y. Dodge and D. Cox. *The Oxford dictionary of statistical terms*. Oxford University Press, USA, 2003.

DGKR18. B. David, P. Gaži, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.

DMNP21. A. Day, E. Medvedev, A. Nirmal, and W. Price. Introducing six new cryptocurrencies in bigquery public datasets—and how to analyze them. *Google Cloud*, 2021.

DNPR10. C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724, 2010.

Dou02. J. R. Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

DR+14. C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

DR16. C. Dwork and G. N. Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

DSBPS+19. S. Delgado-Segura, S. Bakshi, C. Pérez-Solà, J. Litton, A. Pachulski, A. Miller, and B. Bhattacharjee. Txprobe: Discovering bitcoin's network topology using orphan transactions. In *International Conference on Financial Cryptography and Data Security*, pages 550–566. Springer, 2019.

Dwo10. C. Dwork. Differential privacy in new settings. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 174–183. SIAM, 2010.

Eth22. Ethscan. Ethscan Statistics. `https://ethscan.org/`, 2022.

fas21. Fast approximation of natural log. https://gist.github.com/LingDong-/7e4c4cae5cbbc44400a05fba65f06f23, 2021.

Fou22. E. Foundation. *Ethereum 2.0*. 2022.

FRPU94. U. Feige, P. Raghavan, D. Peleg, and E. Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994.

GKZ19. P. Gaži, A. Kiayias, and D. Zindros. Proof-of-stake sidechains. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 139–156. IEEE, 2019.

GN17. Y. Gao and H. Nobuhara. A proof of stake sharding protocol for scalable blockchains. *Proceedings of the Asia-Pacific Advanced Network*, 44(1):13–16, 2017.

GOT19. C. Ganesh, C. Orlandi, and D. Tschudi. Proof-of-stake protocols for privacy-aware blockchains. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 690–719. Springer, 2019.

Gro06. J. Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 444–459. Springer, 2006.

Gro16. J. Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 305–326. Springer, 2016.

GS97. C. M. Grinstead and J. L. Snell. *Introduction to probability*. American Mathematical Soc., 1997.

HMA$^+$17. S. Haney, A. Machanavajjhala, J. M. Abowd, M. Graham, M. Kutzbach, and L. Vilhuber. Utility cost of formal privacy for releasing national employer-employee statistics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1339–1354, 2017.

HR70. M. Hellman and J. Raviv. Probability of error, equivocation, and the chernoff bound. *IEEE Transactions on Information Theory*, 16(4):368–372, 1970.

KK07. R. M. Karp and R. Kleinberg. Noisy binary search and its applications. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 881–890, 2007.

KKKZ19. T. Kerber, A. Kiayias, M. Kohlweiss, and V. Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 157–174. IEEE, 2019.

KM14. D. Kifer and A. Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Transactions on Database Systems (TODS)*, 39(1):1–36, 2014.

KMNS21. M. Kohlweiss, V. Madathil, K. Nayak, and A. Scafuro. On the anonymity guarantees of anonymous proof-of-stake protocols. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1818–1833. IEEE, 2021.

KN12. S. King and S. Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19(1), 2012.

KOV15. P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.

KPXP14. G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12):1155–1166, 2014.

KRDO17. A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

KS69. O. Krafft and N. Schmitz. A note on hoeffding's inequality. *Journal of the American Statistical Association*, 64(327):907–912, 1969.

LSV$^+$19a. M. Lécuyer, R. Spahn, K. Vodrahalli, R. Geambasu, and D. Hsu. Privacy accounting and quality control in the sage differentially private ml platform. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, page 181–195, New York, NY, USA, 2019. Association for Computing Machinery.

LSV$^+$19b. M. Lécuyer, R. Spahn, K. Vodrahalli, R. Geambasu, and D. Hsu. Privacy accounting and quality control in the sage differentially private ml platform. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 181–195, 2019.

Maz15. D. Mazieres. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 32:1–45, 2015.

MLP$^+$15. A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee. Discovering bitcoin's public topology and influential nodes. *et al*, 2015.

NB08. S. Nakamoto and A. Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 4, 2008.

NFPH15. A. Narayan, A. Feldman, A. Papadimitriou, and A. Haeberlen. Verifiable differential privacy. In *Proceedings of the Tenth European Conference on Computer Systems*, pages 1–14, 2015.

NM$^+$16. S. Noether, A. Mackenzie, et al. Ring confidential transactions. *Ledger*, 1:1–18, 2016.

OM14. K. J. O'Dwyer and D. Malone. Bitcoin mining and its energy footprint. 2014.

Ros14. S. M. Ross. *Introduction to probability models*. Academic press, 2014.

SCG$^+$14. E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

Shi11. S. A. Shirali. The bhaskara-aryabhata approximation to the sine function. *Mathematics Magazine*, 84(2):98–107, 2011.

Taw05. S. A. Tawfik. Minimax approximation and remez algorithm. *Lecture Notes at http://www. math. unipd. it/~ alvise/CS_2008/APPROSSIMAZIONE_2009/MFILES/Remez. pdf*, 2005.
VJR18. D. Vujicic, D. Jagodic, and S. Randic. Blockchain technology, bitcoin, and ethereum: A brief overview. In *2018 17th international symposium infoteh-jahorina (infoteh)*, pages 1–6. IEEE, 2018.
W+14. G. Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.
WBNM21. C. Wang, J. Bater, K. Nayak, and A. Machanavajjhala. Dp-sync: Hiding update patterns in secure outsourced databases with differential privacy. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1892–1905, 2021.
WBNM22. C. Wang, J. Bater, K. Nayak, and A. Machanavajjhala. Incshrink: Architecting efficient outsourced databases using incremental mpc and differential privacy. *arXiv preprint arXiv:2203.05084*, 2022.
Woo16. G. Wood. Polkadot: Vision for a heterogeneous multi-chain framework. *White Paper*, 21:2327–4662, 2016.
XLCB18. B. Xu, D. Luthra, Z. Cole, and N. Blakely. Eos: An architectural, performance, and economic analysis. *Retrieved June*, 11:2019, 2018.
Zam17. V. Zamfir. Casper the friendly ghost: A correct by construction blockchain consensus protocol. *Whitepaper: https://github. com/ethereum/research/blob/master/papers/caspertfg/caspertfg. pdf*, 2017.
zok22. A toolbox for zksnarks on ethereum. `https://github.com/Zokrates/ZoKrates`, 2022.

A UC Model of Private Ledger

In this section, we offer an extended overview of the Universal Composability (UC) formalization of private (distributed) ledgers. For a comprehensive understanding of this topic, we recommend readers refer to important previous works, such as [DGKR18,KRDO17,BGK+18,KKKZ19,KMNS21]. Hence, in what follows, we will focus on reviewing important related contents that enable readers to comprehend the technical intricacies of our study and omit low-level details.

A.1 UC security in Brief

In the UC security framework [Can01], we consider the following entities: (i)*Ideal functionality* $\mathcal{G}$: a trusted third party (cannot be corrupted) that performs specific computations based on parties' commands. (ii) *Honest parties* adhere to a specified protocol, interacting with one another, the environment, and functionalities. (iii) The *adversary* corrupts a set of parties, referred to as dishonest parties, and prompts them to deviate arbitrarily from the protocol. (iv) The *environment* $\mathcal{E}$ is a program that selects inputs for honest parties, interacts with the adversary (representing corrupted parties), and receives outputs from all parties.

UC incorporates ideal and real execution models, with the actual protocol π operating in the real model. In the ideal model, honest parties act primarily as relayers, conveying information between their environment $\mathcal{E}$ and the trusted functionality $\mathcal{G}$. Security requires that for any adversary $\mathcal{A}$ in the real model, there should exist a simulator $\mathcal{S}$ that corrupts the same set of parties as $\mathcal{A}$ and renders the ideal-world execution indistinguishable from the real-world executions from the perspective of any environment $\mathcal{E}$. More specifically, let $\mathsf{IDEAL}^{\mathcal{G}}_{\mathcal{S},\mathcal{E}}(\lambda, z)$, and $\mathsf{REAL}^{\pi}_{\mathcal{A},\mathcal{E}}(\lambda, z)$, to be the transcript of the ideal world execution, respectively, where λ is the security parameter and $z \in \{0, 1\}^*$ is the input environment.

Theorem 19 (UC-emulation). *Protocol π UC-emulates functionality $\mathcal{G}$ if for all adversaries $\mathcal{A}$, there exists an simulator $\mathcal{S}$ (who corrupts the same parties as $\mathcal{A}$) such that the following holds*

$$\begin{aligned} &\{\mathsf{IDEAL}^{\mathcal{G}}_{\mathcal{S},\mathcal{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \\ \approx_{\mathsf{ind}} &\{\mathsf{REAL}^{\pi}_{\mathcal{A},\mathcal{E}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \end{aligned} \tag{4}$$

For ease of notation, we may abuse the notation $\mathsf{IDEAL}^{\mathcal{G}}_{\mathcal{S},\mathcal{E}}$, and $\mathsf{REAL}^{\pi}_{\mathcal{A},\mathcal{E}}$ for the aforementioned assembles in Equation 4 and omits the inputs.

A.2 Private ledger functionality $\mathcal{G}_{\mathsf{PL}}$

Next, we provide the description of the private ledger functionality, denoted as $\mathcal{G}_{\mathsf{PL}}$. This is primarily derived from the functionalities explored in [KKKZ19,KMNS21], which in turn expanded on earlier research on distributed ledgers [KRDO17,DGKR18]. In accordance with the construction outlined in [KKKZ19,KMNS21], we consider in the ideal model, the simulator $\mathcal{S}$ can read a sequence of honest inputs, denoted as $\mathcal{I}_H$, which also encompass blinded versions of transactions submitted by honest parties. The simulator can also query a general purpose leakage profile, Lkg that (i) assists in simulating the protocol's leadership election, or in other words, helps determine which party has been activated by $\mathcal{E}$ to perform ledger maintenance activities; and (ii) allows the simulator to know when corrupted parties submitted transactions get confirmed. We assume that parties can access a global clock functionality, represented by $\mathcal{G}_{\mathsf{clock}}$. This functionality maintains a record of the current (global) round and provides this information to any party that requests about it.

Functionality 1.1 shows the details of the $\mathcal{G}_{\mathsf{PL}}$. To ensure privacy, $\mathcal{G}_{\mathsf{PL}}$ conceals transactions details by only allow adversary to read the blind version of transactions submitted by honest parties.

Functionality 1.1: $\mathcal{G}_{\mathsf{PL}}$

$\mathcal{G}_{\mathsf{PL}}$ is parameterized by a general purpose leakage function Lkg. $\mathcal{G}_{\mathsf{PL}}$ manages: (i) a ledger state, state; (ii) a buffer of unconfirmed transactions, buffer; (iii) a sequence of honest inputs, $\mathcal{I}_H$; and (iv) a pointer ptr_i for each party P_i represents its local state. i.e. the length of the prefix of state that is tailored to P_i. We use $\overrightarrow{\mathsf{ptr}}$ to denote of all parties' local stake pointer.

In what follows, we will denote the set of honest parties as H, and the set of all registered parties as P.

Upon receiving any input I from party P_i, consult $\mathcal{G}_{\mathsf{clock}}$ to retrieve current time t, and record interaction in $\mathcal{I}_H$. Specifically, if I is not `SUBMIT` command, then $\mathcal{I}_H \leftarrow \mathcal{I}_H || (I, P_i, t)$, then evaluate the following:

(a) (Add Transaction). If I is $(\mathtt{SUBMIT}, \mathsf{sid}, \mathsf{tx})$, add tx to buffer if the transaction is valid, update $\mathcal{I}_H = \mathcal{I}_H || (I, \mathsf{blind}(\mathsf{tx}), t)$, and reveal $\mathsf{blind}(\mathsf{tx})$ to $\mathcal{A}$, where $\mathsf{blind}(\mathsf{tx})$ hides transaction details, i.e., the sender, recipient, and the amount of tx.

(b) (Read State). If I is $(\mathtt{READ}, \mathsf{sid})$, send back the blind transactions, $\mathcal{I}_H$, and the leakage Lkg. Set $\mathsf{state}_i = \mathsf{state}[\mathsf{ptr}_i]$, and return $\mathsf{blind}(\mathsf{state}_i)$, the blinded version of P_i's local state, back to P_i. If $P_i \in P/H$, send $(\mathsf{state}_i, \mathcal{I}_H, \mathsf{Lkg})$ to the adversary.

(c) (Extend chain). If I is $(\mathtt{MAINTAIN\_LEDGER}, \mathsf{sid})$, then perform the ledger maintenance actions. For the low-level details of the consensus, we refer readers to the important related works [KRDO17,DGKR18]. In general, the goal is to continuously update state with valid transactions in buff. At the same time $\mathcal{G}_{\mathsf{PL}}$ ensures the following properties:
   - The update of state is append only, and transactions are typically added to state by block structures. This can be represented as: $\mathsf{state} \leftarrow \mathsf{state}||(\mathsf{Block}(\mathsf{tx}_1, \mathsf{tx}_2, ..., \mathsf{tx}_\mathsf{m}), t)$.
   - Within bounded delay, valid transactions in buff will be added to state †.
   - We consider the adversary can set slackness for each party's stake pointer, however, for all pointers the maximum lag behind the most current state is bounded by a certain value, i.e. $\forall\ i\ |\mathsf{state}| - |\mathsf{state}[\mathsf{ptr}_i]| \le \mathsf{lag\_bound}$

† In terms of randomized protocols, this property may be ensured under certain (high) probability

In $\mathcal{G}_{\mathsf{PL}}$, we abuse the notation $\mathsf{blind}(\cdot)$ for both the blind version of transactions submitted by honest parties and the blinded local state. For blind transactions, $\mathsf{blind}(\mathsf{tx})$, we assume that the sender, recipient, and amount are all concealed. In the case of the blinded local state, i.e., $\mathsf{blind}(\mathsf{state}_i)$, it replaces all transactions in state_i that are not submitted by party P_i with blind transactions while leaving the others unaltered. In other words, the adversary can still determine when transactions submitted by corrupted parties are confirmed and the details of those transactions. Beyond that, the adversary's knowledge is bounded to a sequence of blind transactions.

A.3 Leakage profile in $\mathcal{G}_{\mathsf{PL}}$

The functionality is parameterized with a leakage function, Lkg, which essentially simulates the protocol's leadership election and reveals the winning party. This is necessary because, otherwise, the simulator would be unable to identify which party is activated with the `MAINTAIN_LEDGER` command. Furthermore, Kohlweiss et al. [KMNS21] provided a theoretical proof that no simulator exists if the ideal functionality does not disclose the block proposers' identities, even if parties have the access to an idealized functionality for anonymous broadcasting. We note that this leakage is indeed protocol-specific, however, as all PoS blockchains consider to elect leaders proportional to their owned stake, thus one may formulate Lkg as a (possibly probabilistic) function $\mathsf{Lkg} = f(\{S^i\}_{1\le i\le n})$, such that $S^i = \{s^i_j\}_{j\in\mathbb{N}^+}$ denotes party P_i's entire stake history at each step.

In what follows we provide an example leakage profile based on the leader election protocol of Ouroboros Genesis [BGK+18]. Note that Ouroboros Genesis itself does not consider transaction privacy. Thus by default, the ledger reveals its entire content to the simulator, such as state, buff, etc. Here, we only adopt the leader election design from Ouroboros Genesis and create a sample leakage profile accordingly.

Example 20 (Example leakage $\mathsf{Lkg}_{\mathsf{Gen}}$*).* Let $P\{P_1, ..., P_n\}$ to be all parties, S^i as party P_i's stake history. For every time t, the odds that P_i is elected as a block proposer is determined by a slot leader election function $\phi(s^i_t) \in (0, 1)$ over P_i's stake at time t. Then leakage $\mathsf{Lkg}_{\mathsf{Gen}}$ is abstract in Algorithm 2

Algorithm 2 $\mathsf{Lkg}_{\mathsf{lead}}(\{S^i\}_{1\le i\le n}, t)$

```
1: Initiate the leader set $\mathsf{Lead} = \emptyset$
2: for $j = 1, 2, 3, ..., t$ do
3:     Initiate $\mathsf{Lead}_j = \emptyset$
4:     for $i = 1, 2, 3, ..., n$ do
5:         Add $P_i$ to $\mathsf{Lead}_j$ with probability $\phi(s^i_j)$
6:     $\mathsf{Lead} \leftarrow \mathsf{Lead} \cup \mathsf{Lead}_j$
7: return $\mathsf{Lead}$
```

B Private PoS with Stake Distortion

B.1 Standard ledger maintenance

Based on the private ledger functionality $\mathcal{G}_{\mathsf{PL}}$, the command (MAINTAIN_LEDGER, sid) triggers the main ledger update activities. In a standard protocol implementation, i.e. [LSV+19b,KMNS21,GOT19], the execution of ledger maintenance is separated into distinct time slots, during which one or more slot leaders are chosen for each slot to modify the ledger states. In PoS blockchains, slot leaders are selected according to each party's owned stake. More specifically, for every round, each party retrieves all relevant information for the current round from the network, obtains the most recent stake parameters, then initiates a private lottery such that the winning odd is proportional to her stake. Depending on the consensus, the winner maintains the ledger in different ways. For instance, in the longest chain-based PoS [BGK+18,W+14], the winner maintains the ledger (i.e., extends the chain with new blocks) for a predefined time slot. While for committee-based PoS [CM16], a lottery winner wins the right to take part in committee(s) and one or more of the members determine the next block.

B.2 Stake distortion functionality

We provide the formal functionality descriptions for both the timer ($\mathcal{F}_{\mathsf{Timer}}$) and the binary ($\mathcal{F}_{\mathsf{Bin}}$) mechanism.

Timer mechanism $\mathcal{F}_{\mathsf{Timer}}$. The details are provided in Functionality 2.1. In general, parties can interact with $\mathcal{F}_{\mathsf{Timer}}$ with command DISTORT to retrieve the distorted stake at each time, or with command GET_COMM to obtain distorted stake commitments for every other party.

Functionality 2.1: $\mathcal{F}_{\mathsf{Timer}}$

```
    For every time step $j$ evaluate the following:
1:  if $j \mod T = 0$ then
2:      for party $i = 1, 2, ..$ do
3:          Determine the true stake $s^i_j$ owned by $P_i$
4:          Distort $P_i$'s true stake using Laplace noise: $\tilde{s}^i_j \leftarrow s^i_j + \mathsf{Lap}(\frac{\alpha}{\epsilon})$
5:          Commit $\tilde{s}^i_j$ to $\mathsf{com}_{\tilde{s}^i_j}$ with $r^i_j$
6:          Record $\tilde{s}^i_j$, $\mathsf{com}_{\tilde{s}^i_j}$, and $r^i_j$ for $P_i$
7:  else
8:      for party $i = 1, 2, ..$ do
9:          Reuse the objects from last time step: $\tilde{s}^i_j \leftarrow \tilde{s}^i_{j-1}$, $r^i_j \leftarrow r^i_{j-1}$, $\mathsf{com}_{\tilde{s}^i_j} \leftarrow \mathsf{com}_{\tilde{s}^i_{j-1}}$
10:         Record $\tilde{s}^i_j$, $\mathsf{com}_{\tilde{s}^i_j}$, and $r^i_j$ for $P_i$

    Upon receiving (DISTORT, sid) from $P_i$ :
11: return $\tilde{s}^i_j$, $\mathsf{com}_{\tilde{s}^i_j}$, and $r^i_j$.
12: add $\mathsf{com}_{\tilde{s}^i_j}$ to every registered party's local cache.
    Upon receiving (GET_COMM, id, sid) from $P_i$ :
13: return $\mathsf{com}_{\tilde{s}^k_j}$ from $P_i$'s cache
```

Binary (tree) mechanism $\mathcal{F}_{\mathsf{Bin}}$. The details are provided in Functionality 2.2. Parties can interact with $\mathcal{F}_{\mathsf{Bin}}$ with command `DISTORT` to retrieve the distorted stake at each time, or with command `GET_COMM` to obtain distorted stake commitments for every other party along with the commitments to the p-sums and noisy p-sums that derives the distorted stake.

Functionality 2.2: Binary mechanism $\mathcal{F}_{\mathsf{Bin}}$

```
We consider the functionality maintains a binary interval tree $\mathsf{IT}^i$ for every party $P_i$.

For every time step $j$ evaluate the following:
1: if $j \mod L = 0$ then
2:     for party $i = 1, 2, ..$ do
3:         Determine the true stake $s^i_j$ owned by $P_i$
4:         Distort $P_i$'s true stake using Laplace noise: $\tilde{s}^i_j \leftarrow s^i_j + \mathsf{Lap}(\frac{\alpha}{\epsilon})$
5:         Commit $\tilde{s}^i_j$ to $\mathsf{com}_{\tilde{s}^i_j}$ with $r^i_j$
6:         Record $\tilde{s}^i_j$, $\mathsf{com}_{\tilde{s}^i_j}$, and $r^i_j$ for $P_i$
7:         Cache $\mathsf{tx}^* = \tilde{s}^i_j$, $t^* = j$ for $P_i$
8: if $(t \leftarrow j \mod L) \mod T = 0$ then
9:     for party $i = 1, 2, ..$ do
10:        Initiate $ct^i = \emptyset$
11:        $a \leftarrow t^*, b \leftarrow 0$
12:        while $a \neq j$ do
13:            Get node $[a, b] \in \mathsf{IT}^i$ such that:
                   $|a - b| = 2^k T$, and $2^{k+1} T > |j - a|$
14:            $ct^i \leftarrow ct^i \cup [a, b]$
15:            $a \leftarrow b + 1$
16:        for $k = 1, 2, 3, ...$ do
17:            if $\tilde{ct}^i[k]$ exists then
   //$\tilde{ct}^i[k]$ denotes the noisy p-sum of $ct^i[k]$
18:                read $\tilde{ct}^i[k]$ from cache
19:            else
20:                $\tilde{ct}^i[k] \leftarrow ct^i[k] + \mathsf{Lap}(\frac{\alpha}{\epsilon})$
21:                Commit $\tilde{cv}^i[k]$ to $\mathsf{com}_{\tilde{cv}^i[k]}$
22:        $\tilde{s}^i_j \leftarrow \sum_k \tilde{ct}[k] + \mathsf{tx}^*$
23:        Commit $\tilde{s}^i_j$ to $\mathsf{com}_{\tilde{s}^i_j}$ with $r^i_j$
24:        Record $\tilde{s}^i_j$, $\mathsf{com}_{\tilde{s}^i_j}$, and $r^i_j$ for $P_i$
25: else
26:     for party $i = 1, 2, ..$ do
27:         $\tilde{s}^i_j \leftarrow \tilde{s}^i_{j-1}$, $r^i_j \leftarrow r^i_{j-1}$, $\mathsf{com}_{\tilde{s}^i_j} \leftarrow \mathsf{com}_{\tilde{s}^i_{j-1}}$
28:         Record $\tilde{s}^i_j$, $\mathsf{com}_{\tilde{s}^i_j}$, and $r^i_j$ for $P_i$
   //$\mathsf{com}_{ct^k}$ and $com_{\tilde{ct}^k}$ represent the commitments to the p-sums and noisy p-sums for computing $\tilde{s}^i_j$
   Upon receiving $(\mathtt{DISTORT}, \mathsf{sid})$ from $P_i$ :
29: return $\tilde{s}^i_j$, $\mathsf{com}_{\tilde{s}^i_j}$, and $r^i_j$
30: add $\mathsf{com}_{\tilde{s}^i_j}$, $\mathsf{com}_{ct^k}$, $\mathsf{com}_{\tilde{ct}^k}$ to every registered party's local cache.
   Upon receiving $(\mathtt{GET\_COMM}, \mathsf{id} = k, \mathsf{sid})$ from $P_i$ :
31: return $\mathsf{com}_{\tilde{s}^k_j}$, $\mathsf{com}_{ct^k}$, $\mathsf{com}_{\tilde{ct}^k}$ from $P_i$'s cache
```

B.3 Security proof

In this section, we present the primary security proof (for Theorem 18) related to the stake distortion protocol proposed in this work. To minimize redundancy, we focus on the security proof for the Timer protocol. However, we emphasize that using the same proof technique, one can easily derive a proof for the Binary protocol.

Proof. (**Theorem 18**)We prove this theorem by constructing a simulator $\mathcal{S}_{\mathsf{timer}}$, which corrupts the same set of nodes as the real-world adversary $\mathcal{A}$ and interacts with $\mathcal{F}_{\mathsf{sd}}$. This simulator can produce a transcript that is computationally indistinguishable from the real-world protocol execution of $\prod_{\mathsf{sd}}$ in the presence of $\mathcal{A}$. We assume the existence of a simulator that simulates the message broadcast channel (the network) as described in [GOT19]. Consequently, we will not delve into the details of simulating message broadcasting in real-world execution. For more information, readers may refer to [GOT19].

To construct the simulator, we consider the existence of a simulator, $\mathcal{S}_{\mathsf{nizk}}^{\mathcal{L}_{\mathsf{timer}}}$, for non-interactive zero-knowledge (NIZK) that proves statements in the NP language of $\mathcal{L}_{\mathsf{timer}}$. $\mathcal{S}_{\mathsf{nizk}}^{\mathcal{L}_{\mathsf{timer}}}$ is able to simulate a proof using a trapdoor (without requiring the witness) that is indistinguishable from one an honest prover would provide with the witness [Bel20,Gro06]. We refer to the proof generated by $\mathcal{S}_{\mathsf{nizk}}^{\mathcal{L}_{\mathsf{timer}}}$ using the trapdoor as the simulated proof. The detailed construction is provided in Simulator 2.1.

Simulator 2.1: Simulator $\mathcal{S}_{\mathsf{timer}}$

```
    Simulating noise generation keys:
 1: for $i = 1, 2, 3...$ do
 2:     if $P_i$ is a dishonest party then
 3:         return $P_i$'s noise keys $(\mathsf{n}^i_{\mathsf{pk}}, \mathsf{n}^i_{\mathsf{sk}})$.
 4:     else
    //simulate honest party's private key using a random string
 5:         Sample a random string as $\mathsf{n}^i_{\mathsf{pk}}$
 6:         return $\mathsf{n}^i_{\mathsf{pk}} \leftarrow \mathsf{PRF}_{\mathsf{n}^i_{\mathsf{sk}}}(0)$

    Simulating DP noises:
 7: for $i = 1, 2, 3...$ do
 8:     if $P_i$ is a dishonest party then
 9:         Determine the public randomness $\mathsf{ra}, \mathsf{rb}$.
    //Internally emulates $\mathcal{F}_{\mathsf{Timer}}$ to produce DP noises
10:         $z^0 = \mathsf{PRF}_{\mathsf{n}^i_{\mathsf{sk}}}(\mathsf{ra}), z^1 = \mathsf{PRF}_{\mathsf{n}^i_{\mathsf{sk}}}(\mathsf{rb})$
11:         return $z \leftarrow \frac{\alpha}{\epsilon}(\log(z^0) - \log(z^1))$
12:     else
    //simulate honest party's DP noise with simulated randomness and keys
13:         Sample random strings $\mathsf{r}_1, \mathsf{r}_2$.
14:         $z^0 = \mathsf{PRF}_{\mathsf{n}^i_{\mathsf{sk}}}(\mathsf{r}_1), z^1 = \mathsf{PRF}_{\mathsf{n}^i_{\mathsf{sk}}}(\mathsf{r}_2)$
15:         return $z \leftarrow \frac{\alpha}{\epsilon}(\log(z^0) - \log(z^1))$

    Simulating $\mathcal{F}_{\mathsf{stk}}$:
16: for $i = 1, 2, 3...$ do
17:     if $P_i$ is a dishonest party then
18:         Determine $P_i$'s current stake $s^i$.
19:         Sample $r^i$ and $\mathsf{com}_{s^i} \leftarrow \mathsf{Com}_{r^i}(s^i)$
20:     else
    //simulate honest party's commitment with 0
21:         Sample $r^i$ and $\mathsf{com}_{s^i} \leftarrow \mathsf{Com}_{r^i}(0)$

22: Upon receiving a request from (dishonest) $P_i$ to retrieve distorted stake: return $s^i, \mathsf{com}_{s^i}, r^i$ to $P_i$
23: Upon receiving a request from (dishonest) $P_i$ to retrieve stake commitments for honest party with $\mathsf{pid} = k$:
    return $\mathsf{com}_{s^k}$ to $P_i$
24: Upon receiving a request from (dishonest) $P_i$ to retrieve noise generation key for honest party with
    $\mathsf{pid} = k$: return the simulated key $\mathsf{n}^k_{\mathsf{pk}}$.
    Simulating noisy stake (and commitments):
25: for $i = 1, 2, 3...$ do
26:     if $P_i$ is a dishonest party then
27:         Internally emulates $\mathcal{F}_{\mathsf{Timer}}$ to derive $\tilde{s}^i$, $\mathsf{com}_{\tilde{s}^i}$, and the opening $\tilde{r}^i$ to $\mathsf{com}_{\tilde{s}^i}$.
28:         Generate NIZK proof $\pi^i$ with witness.
29:     else
    //simulate noisy stake and commitments using true stake=0, simulated noise $z$ and random opening
30:         Simulate noisy stake $\tilde{s}^i$ using the simulated DP noise and true stake as 0.
31:         Sample $\tilde{r}^i$, $\mathsf{com}_{\tilde{s}^i} = \mathsf{Com}_{\tilde{r}^i}(\tilde{s}^i)$
32:         Interact with $\mathcal{S}^{\mathcal{L}_{\mathsf{timer}}}_{\mathsf{nizk}}$ to extract simulated proof $\pi^i$ without witness.

33: Upon receiving a request from (dishonest) $P_i$ to retrieve distorted stake: return $\tilde{s}^i$, $\tilde{r}^i$, to $P_i$ and simulate
    message broadcasting for $\mathsf{com}_{\tilde{s}^i}$, $\pi^i$.
34: Upon receiving a request from (dishonest) $P_i$ to retrieve stake commitments for honest party with $\mathsf{pid} = k$:
    return $\mathsf{com}_{\tilde{s}^k}$, $\pi^k$ to $P_i$.
```

Let HYB_0 represent the distribution of real-world protocol execution of $\prod_{\mathsf{timer}}$ (in the hybrid world where auxiliary functionalities are available). We consider the world HYB_1, which is the same as the protocol execution, except for the following: the noise generation keys and the DP noises are obtained from the simulator $\mathcal{S}_{\mathsf{timer}}$. The simulator $\mathcal{S}_{\mathsf{timer}}$ generates the keys and DP noises using the corresponding simulation primitives. Since the private noise generation key, n_{sk} itself is a random string, thus by the security of PRF the output of $\mathsf{PRF}_{\mathsf{n}_{\mathsf{sk}}}(0)$ is computational indistinguishable from $\mathsf{PRF}_{\mathsf{rd\_string}}(0)$. Moreover, by the fundamental transformation laws of probabilities, Laplace or Gaussian random variables are statistically indistinguishable from the ones transformed from uniform distributions. Hence, we say that the distributions of HYB_0 and HYB_1 are indistinguishable.

Next, we consider another world HYB_2, which is identical to the world HYB_1 except for replacing the calls to $\mathcal{F}_{\mathsf{stk}}$ with the interaction to $\mathcal{S}_{\mathsf{timer}}$. The simulator $\mathcal{S}_{\mathsf{timer}}$ generates necessary outputs and responds to requests using the appropriate simulation primitives. In HYB2, the simulator simulates the honest party's stake commitment using $\mathsf{Com}r(0)$, where r is a random opening, and returns the honest party's public noise generation key with the simulated one (as demonstrated in HYB_1, the simulated n_{pk} is indistinguishable from the actual key). Owing to the equivocality property of commitment schemes [KKKZ19], we can conclude that the distributions of HYB_1 are indistinguishable from those of HYB_2.

Lastly, we consider the ideal world HYB3. The only difference between HYB3 and HYB2 is that in HYB3, the noisy stake commitments for honest parties are simulated using (i) 0 stake as the true value; (ii) simulated DP noise z; (iii) a random opening to derive the commitment; and (iv) a simulated NIZK proof generated by $\mathcal{S}^{\mathcal{L}_{\mathsf{timer}}}\mathsf{nizk}$. Due to the equivocality property of commitments, we know that the simulated noisy stake commitments for honest parties are indistinguishable from the actual ones. Furthermore, the simulation security of NIZKs (or zero-knowledge property) ensures that the simulated proof by $\mathcal{S}^{\mathcal{L}_{\mathsf{timer}}}\mathsf{nizk}$, generated without using a witness, is computationally indistinguishable from the actual proof obtained from $\mathcal{F}^{\mathcal{L}}_{\mathsf{nizk}}$. Consequently, we conclude that the distribution of the ideal world HYB_3 is indistinguishable from that of HYB_2 and, by extension, also indistinguishable from the real-world executions, HYB_0.

One can employ similar proof techniques to establish the security of the Binary protocol. To simulate the execution of $\prod_{\mathsf{Bin}}$, the simulator would also need to simulate the commitments to the p-sums and noisy p-sums for honest parties. However, this can be achieved using a similar approach, such as setting all p-sums to 0 and generating commitments with random openings.

C Proof of Theorems

C.1 Stake Inference Attacks

In this section, we provide proofs regarding theorems for the proposed stake inference attacks (Section 3).

Proof of Theorem 6 We start this section by introducing the necessary Lemma 21 that is used in our main proof. We then prove the bounded failure probability of Algorithm 1 in Theorem 22 and the running time in Theorem 23.

Lemma 21 (Hoeffding's Inequality [KS69]). *Given a biased coin with heads probability p. Let p_n denote the heads frequency for n independent coin flips. For any $n > 1$ and $\tau > 0$, the heads frequency p_n satisfies*

$$\Pr\left[|p_n - p| \geq \tau\right] \leq e^{-2\tau^2 n}$$

Theorem 22. *Given $\delta \in (0,1)$, with probability at least $1-\delta$, $p\mathsf{SC}(...,\delta)$ outputs correct comparisons.*

Proof. We first prove that for each round, the probability that Algorithm 1 outputs the wrong comparison between the two biased coins $q_(f_v)$ and $q(f_{\mathsf{cmp}})$ is at most δ_i. Without loss of generality, we consider the case where the algorithm outputs $f_v > f_{\mathsf{cmp}}$ ($p_0 > p_1$ and $|p_0 - p_1| > 2\tau_i$). By Lemma 21, and the union bound, the failure probability that Algorithm 1 outputs the wrong comparison satisfies

$$\begin{aligned}&\Pr[\mathsf{wrong\_cmp}]\\&\leq \Pr[q(f_v) < p_0 - \tau_i] + \Pr\left[q(f_{\mathsf{cmp}}) > p_1 + \tau_i\right]\\&\leq e^{-2}e^{\log(\frac{1}{\delta_i})} = e^{-2}\delta_i\end{aligned} \tag{5}$$

Next, by combing union bound, we show that the total failure probability is bounded by δ, which is sufficient to prove $\sum_i^{+\infty}\delta_i < \delta$. Given that

$$e^{-1} = \int_0^{e^{-1}} 1\,dx = \sum_{n=1}^{+\infty}(e^{-n} - e^{-(n+1)}) = (1-e^{-1})\sum_{n=1}^{+\infty} e^{-n} \tag{6}$$

Thus by Equation 6, we can obtain

$$\sum_{i=1}\frac{\delta}{e^i} < \delta\sum_{i=1} e^{-i} < \frac{\delta e^{-1}}{1-e^{-1}} = \frac{\delta}{e-1} \tag{7}$$

Theorem 23. *The expected running time of $p\mathsf{SC}$ is bounded by $O\left(\frac{\log(1/\delta)+\log(1/\max(\tau,\tau_{\mathsf{c}}))}{\max(\tau,\tau_{\mathsf{c}})^2}\right)$ (sim) coin flips.*

Proof. We prove the complexity of $p\mathsf{SC}$ by following the same technique used by [KK07] for proving Lemma 3.2. Without loss of generality, we consider $q(f_v) > q(f_{\mathsf{cmp}})$.

We start with the case of $\tau_c > \tau$, and let $k = \log(\frac{1}{\tau_c})$, so for any round $\ell > k$ we have $\tau_\ell < \frac{\tau_c}{2}$. In order to let the algorithm keep running after $\ell > k$ rounds, then at least one of the following must be true

$$\left(p_0^{(\ell)} < q(f_v) - \frac{\tau_c}{2}\right) \text{ or } \left(p_1^{(\ell)} > q(f_{\mathsf{cmp}}) + \frac{\tau_c}{2}\right)$$

where $p_0^{(\ell)}$, and $p_1^{(\ell)}$ denotes the estimated heads frequency for $q(f_v)$, and $q(f_{\mathsf{cmp}})$, respectively under round ℓ. By Lemma 21, the probability is at most

$$e^{-\frac{\tau_c^2 n}{2}} \leq e^{-\frac{\tau_c^2\log(\frac{1}{\delta_\ell})}{2\tau_\ell^2}} = \delta\cdot e^{-\left(\frac{e^{2\ell}\tau_c^2}{2}+\ell\right)} \tag{8}$$

Note that the running time of round $\ell > k$ grows exponentially in ℓ, while by Equation 8, the probability that Algorithm 1 to continue running after round $\ell > k$ decrease faster than exponential in ℓ. Hence, the expected running time is bounded by the running time of k^{th} round:

$$O\left(\log\left(\frac{1}{\delta_k}\right)\frac{1}{\tau_c^2}\right) = O\left(\frac{\log\left(\frac{1}{\delta}\right)+\log\left(\frac{1}{\tau_c}\right)}{\tau_c^2}\right) \tag{9}$$

Next, we consider the case where $\tau_c < \tau$. With the same technique, one can obtain that the expected running time is bounded by the running time of k^{th} round where $k = \log(\frac{1}{\tau})$, which is $O\left(\frac{\log(1/\delta)+\log(1/\tau)}{\tau^2}\right)$.

Combining both cases, one can obtain that the expected running time of $p\mathsf{SC}$ is bounded by

$$O\left(\frac{\log(1/\delta)+\log(1/\max(\tau,\tau_{\mathsf{c}}))}{\max(\tau,\tau_{\mathsf{c}})^2}\right) \tag{10}$$

Proof of Theorem 7 We first prove the (expected) running time of RdBin in Theorem 24, then prove the error bound stated in Theorem 25.

Theorem 24. *Let $n = \frac{S}{\theta}$, and $\delta = O(\frac{1}{\log n})$, then the running time of RdBin is bounded by*

$$O\left(\frac{\log(\log n/\tau)}{\tau^2}\times\log n\right)$$

Proof. Let $\delta = \frac{c}{\log n}$, where c is considered to be a constant factor and $c > 0$. By Equation 10, we can obtain that for each random walk phase of RdBin, the expected running time is bounded by

$$O\left(\frac{\log(\log n/c) + \log(1/\tau)}{\tau^2}\right) \sim O\left(\frac{\log(\log n/\tau)}{\tau^2}\right)$$

Note that the random walk will terminate within $\log n$ steps, and thus we can obtain the overall running time of RdBin is bounded by the term $O\left(\frac{\log(\log n/\tau)}{\tau^2} \times \log n\right)$.

Theorem 25. *Let $p_v = q(f_v)$, $\eta = \max\left(q^{-1}(p_v \pm \tau)\right)$, and $\delta = O(\frac{1}{\log n})$, then (i) the random walk of* RdBin *exits at the correct node with constant failure probability; and (ii) the inference error is bounded by* $\max(\frac{\theta}{S}, \eta)$.

Proof. At each step of the random walk, the probability of moving in the wrong direction is bounded by $O(\frac{1}{\log n})$. As such, by union bound, the overall failure probability is bounded by $O(1)$, which is a constant. If the random walk exits correctly, it must have either reached a leaf node or an interval $[f_a, f_b]$ where $|q(\hat{f} = \frac{f_a+f_b}{2}) - q(f_v)| < \tau$, indicating that the minimum comparison threshold has been met. For the first case, since $|f_a - f_b| = \frac{\theta}{S}$, thus the inference error must be bounded by $\frac{\theta}{S}$. For the second case, it is clear that the error must be smaller than $\max\left(|q^{-1}(p_v + \tau) - f_v|, |q^{-1}(p_v - \tau) - f_v|\right)$.

C.2 Privacy Model

We now provide complete proofs for theorems related to our general privacy model (Section 5).

Proof of Theorem 11

Proof. (**Theorem 11**) Consider party P_i and let $S^i = \{s^i_1, s^i_2, ..., s^i_t\}$ to be the stake history of P_i. Given $S_t = \{s_1, s_2, ..., s_t\}$ and $S'_t = \{s'_1, s'_2, ..., s'_t\}$ to be a pair of (α, u)-neighbor stake profiles. Let $x, y \in \mathbb{R}$ such that $x \le y \le x + \alpha$, and by definition of (α, u)-neighbor we have $s_u = x$, $s'_u = y$. We also consider a given output $o \subset Range(\mathsf{Lkg})$ and next. we compute the following probabilities (Note that we simplify the term $\Pr\left[\mathsf{Lkg}^{\langle P_i, S\rangle} = o\right]$ as a conditional probability $\Pr[o \mid S]$)

$$\begin{aligned}\Pr\left[o \mid s^i_u = s_u\right] &= \sum_{S_t \subset \mathbb{S}} \Pr[o \mid S_t] \Pr\left[S_t \mid s^i_u = x\right] \\ &\le \max_{S_t} \Pr[o \mid S_t] \times \sum_{S_t \subset \mathbb{S}} \Pr\left[S_t \mid s^i_u = x\right]\end{aligned} \tag{11}$$

$$\begin{aligned}\Pr\left[o \mid s^i_u = s'_u\right] &= \sum_{S_t \subset \mathbb{S}} \Pr\left[o \mid S'_t\right] \Pr\left[S'_t \mid s^i_u = y\right] \\ &\ge \min_{S'_t} \Pr\left[o \mid S'_t\right] \times \sum_{S'_t \subset \mathbb{S}} \Pr\left[S'_t \mid s^i_u = y\right]\end{aligned} \tag{12}$$

Note that any stake dataset $S \subset \mathbb{S}$ is a (α, u)-neighbor of itself. Thus it holds that $\sum_{S_t \subset \mathbb{S}} \Pr\left[S_t \mid s^i_u = x\right] \le \sum_{S'_t \subset \mathbb{S}} \Pr\left[S'_t \mid s^i_u = y\right]$, since for any S_t there exists at least one (α, u)-neighbor S'_t. In addition, by Definition 10, we know that $\max_{S_t} \Pr[o \mid S_t] \le e^{\omega(t-u)} \times \min_{S'_t} \Pr\left[o \mid S'_t\right]$. Thus by combing the two facts, we can conclude:

$$\begin{aligned}\frac{\Pr\left[o \mid s^i_u = s_u\right]}{\Pr\left[o \mid s^i_u = s'_u\right]} &= \frac{\sum_{S_t \subset \mathbb{S}} \Pr[o \mid S_t] \Pr\left[S_t \mid s^i_u = x\right]}{\sum_{S_t \subset \mathbb{S}} \Pr\left[o \mid S'_t\right] \Pr\left[S'_t \mid s^i_u = y\right]} \\ &\le \frac{\sum_{S_t \subset \mathbb{S}} \Pr\left[S_t \mid s^i_u = x\right]}{\sum_{S'_t \subset \mathbb{S}} \Pr\left[S'_t \mid s^i_u = y\right]} \times \frac{\max_{S_t} \Pr[o \mid S_t]}{\min_{S'_t} \Pr\left[o \mid S'_t\right]} \\ &\le e^{\omega(t-u)} \times \epsilon\end{aligned} \tag{13}$$

By Bayes' theorem

$$\begin{aligned}&\frac{\Pr\left[s^i_u = x \mid o\right]}{\Pr\left[s^i_u = y \mid o\right]} \Bigg/ \frac{\Pr\left[s^i_u = x\right]}{\Pr\left[s^i_u = y\right]} \\ =&\frac{\Pr\left[s^i_u = x \mid o\right] \Pr[o]}{\Pr\left[s^i_u = x\right]} \Bigg/ \frac{\Pr\left[s^i_u = y \mid o\right] \Pr[o]}{\Pr\left[s^i_u = y\right]} \\ =&\frac{\Pr\left[o \mid s^i_u = s_u\right]}{\Pr\left[o \mid s^i_u = s'_u\right]} \le e^{\omega(t-u)} \times \epsilon (\textbf{Definition 8})\end{aligned} \tag{14}$$

In general, Equation 14 shows that for any leakage profile Lkg, if it satisfies Definition 10, then it also satisfies the privacy requirements defined in Definition 8.

C.3 Stake Distortion

We now prove related theorems in Section 6.

Proof of Theorem 13

Proof. (**Theorem** 13) We first recall the important notations: (i) $\mathsf{REAL}^{\pi}_{\mathcal{A},\mathcal{E}}$, denotes the transcript of real world protocol (π) execution between the honest party and the adversary $\mathcal{A}$ involving environment $\mathcal{E}$; (ii) $\mathsf{IDEAL}^{\mathcal{F}}_{\mathcal{S},\mathcal{E}}$, denotes the transcript generated by simulator a $\mathcal{S}$ with input of a leakage profile Lkg who corrupts the same parties as $\mathcal{A}$ and interacts only with a trusted functionality $\mathcal{F}$ and the environment $\mathcal{E}$.

We consider a set of parties $P = \{P_1, P_2, ..., P_n\}$, and the environment $\mathcal{E}$ can activate stake distribution, $\mathbb{S} = \{S^i\}_{1\leq i\leq n}$, for all parties across every time slot. (In fact, the environment $\mathcal{E}$ can activate every transaction for each party, which essentially governs stake distribution among parties over time.)

By definition, as $\prod$ UC-emulates ledger maintenance command (for ease of notation, we denote this as an independent functionality $\mathcal{F}_{\mathsf{LM}}$) in $\mathcal{G}_{\mathsf{PL}}$, thus we know that there must exist a simulator $\mathcal{S}_{\mathsf{LM}}$ such that:

$$\mathsf{IDEAL}^{\mathcal{F}_{\mathsf{LM}}}_{\mathcal{S}_{\mathsf{LM}}(\phi(\mathbb{S})),\mathcal{E}} \approx_{\mathsf{ind}} \mathsf{REAL}^{\Pi(\mathbb{S})}_{\mathcal{A},\mathcal{E}} \tag{15}$$

holds for any environment and any stake distribution.

Next, we prove that with the same simulator $\mathcal{S}_{\mathsf{LM}}$ (for simulating execution transcripts against protocol $\prod$), one can also simulate indistinguishable transcripts against protocol $\prod_{\mathsf{LM}^*}$. We prove this by contradiction.

Assume there exists an environment $\mathcal{E}$ and a distorted stake distribution $\tilde{\mathbb{S}}_a \leftarrow f(\mathbb{S}_a)$ such that

$$\mathsf{IDEAL}^{\mathcal{F}_{\mathsf{LM}}}_{\mathcal{S}_{\mathsf{LM}(\phi(\tilde{\mathbb{S}}_a))},\mathcal{E}} \not\approx_{\mathsf{ind}} \mathsf{REAL}^{\Pi_{\mathsf{LM}^*}(\mathbb{S}_a)}_{\mathcal{A},\mathcal{E}} \tag{16}$$

where $\prod_{\mathsf{LM}^*}(\mathbb{S}_a)$ denotes the execution of protocol $\prod_{\mathsf{LM}^*}$ under the stake distribution $\mathbb{S}_a$. As $\prod_{\mathsf{LM}^*}$ only modifies the inputs rather than the interface with the environment or the components of $\prod$, the equivalence $\mathsf{REAL}^{\Pi_{\mathsf{LM}^*}(\mathbb{S}_a)}_{\mathcal{A},\mathcal{E}} \approx_{\mathsf{ind}} \mathsf{REAL}^{\Pi(\tilde{\mathbb{S}}_a)}_{\mathcal{A},\mathcal{E}}$ becomes evident. In other words, we keep the environment unchanged but only change the stake distribution from $\mathbb{S}_a$ to $\tilde{\mathbb{S}}_a$, then the execution transcripts of $\prod_{\mathsf{LM}^*}(\mathbb{S})$ and $\prod(\tilde{\mathbb{S}})$ should be indistinguishable. And by the assumption Eq 16, we can trivially obtain that

$$\mathsf{IDEAL}^{\mathcal{F}_{\mathsf{LM}}}_{\mathcal{S}_{\mathsf{LM}(\phi(\tilde{\mathbb{S}}_a))},\mathcal{E}} \not\approx_{\mathsf{ind}} \mathsf{REAL}^{\Pi(\tilde{\mathbb{S}}_a)}_{\mathcal{A},\mathcal{E}}$$

This implies that if $\mathcal{E}$ activates the stake distribution as $\tilde{\mathbb{S}}_a$, then simulator $\mathcal{S}_{\mathsf{LM}}$ is unable to generate indistinguishable transcripts against the real protocol executions of $\prod$, which evidently contradicts Equation 15. Consequently, we can deduce that no such environment exists, and the assumption (Eq 16) is incorrect. Therefore, for any stake distribution activation, the same simulator $\mathcal{S}_{\mathsf{LM}}$ can also produce indistinguishable transcripts against the real execution of $\prod_{\mathsf{LM}^*}$ with leakage $\phi(\tilde{\mathbb{S}})$.

Proof of Theorem 14

Proof. (**Theorem** 14) Let $S_t = \{s_j\}_{1\leq j\leq t}$ and $S'_t = \{s'_j\}_{1\leq j\leq t}$ be any (α, u) neighboring stake assignment pairs for an honest party P. We abstract the stake distortion at each time j as $\mathcal{M}_j(s)$, which takes an input stake, s, and outputs $s + \mathsf{Lap}(\frac{\alpha}{\epsilon})$ if $j \mod T == 0$, and $\mathcal{M}_{j-1}(s)$ otherwise. For ease of notation, we write the conditional probability $\Pr\left[\mathcal{M}_j(s_j) = o_j \mid \forall_{1\leq k\leq j-1}\mathcal{M}_k(s_k) = o_k\right]$ as $\Pr\left[\mathcal{M}_j(s_j) = o_j \mid *\right]$, and compute the following term

$$\begin{aligned}\frac{\Pr\left[\tilde{S}_t = \mathbf{o}\right]}{\Pr\left[\tilde{S}'_t = \mathbf{o}\right]} &= \prod_{j=1}^{t} \frac{\Pr\left[\mathcal{M}_j(s_j) = o_j \mid *\right]}{\Pr\left[\mathcal{M}_j(s'_j) = o_j \mid *\right]} \\ &\leq \prod_{j=1}^{u-1} \mathbf{1} \times \prod_{j=u\wedge j=kT,k\in\mathbb{N}^+}^{t} e^{\epsilon} \leq e^{\lfloor\frac{t-u}{T}\rfloor\times\epsilon}\end{aligned} \tag{17}$$

In addition, knowing that $\mathsf{Lkg} = f(\tilde{S}_t)$ is a probabilistic function related to the noisy stake assignment. Thus, by post-processing theorem of DP [DR+14], $\Pr[\mathsf{Lkg} = \mathbf{o}]/\Pr[\mathsf{Lkg}' = \mathbf{o}] \leq e^{\lfloor\frac{t-u}{T}\rfloor\times\epsilon}$. In general, the total privacy loss for stake values at any time u is subject to k-fold composition theorem of DP mechanisms [DR+14], where k denotes the total number of noisy stake releases from time u up to the current moment, which is $\frac{t-u}{T}$.

Proof of Theorem 15

Proof. (**Theorem** 15)

Let $S_t = \{s_j\}_{1 \le j \le t}$ to be the stake profiles of an honest party P, and $\mathsf{tx}_t = \{\mathsf{tx}_j\}_{1 \le j \le t}$ to be the corresponding transaction outcomes at each time, i.e., $\mathsf{tx}_j = s_j - s_{j-1}$. Let $M(\mathsf{tx}_t)$ be a mechanism that processes the noisy p-sums based on tx_t, and $\tilde{S}_t \leftarrow f(M(\mathsf{tx}_t))$ is an algorithm that aggregates the output of M to derive the noisy stake at each time. We say that, for any pair of (α, u)-neighboring stake profiles S_t, and S'_t, with corresponding tx_t, tx'_t. It is obvious that if for any $\mathbf{o} \subseteq Range(M)$,

$$\frac{\Pr[M(\mathsf{tx}_t) = \mathbf{o}]}{\Pr[M'(\mathsf{tx}_t) = \mathbf{o}]} \le e^{\omega(t-u)\times\epsilon} \tag{18}$$

then the released noisy stake $\tilde{S}_t \leftarrow f(M(\mathsf{tx}_t))$ satisfies Definition 10 under the same decay function ω, given f only conduct post-processing operations. By the theory of the Laplace mechanism, $\frac{\Pr[M(\mathsf{tx}_t)=\mathbf{o}]}{\Pr[M'(\mathsf{tx}_t)=\mathbf{o}]} \le e^{\epsilon}$, when $t = u$. Moreover, when $0 < t-u \le L$, then the privacy loss is subject to phase two distortion (privacy loss due to noisy interval tree generation). Note that tx_u can be used to generate at most $\log_2(\frac{(t-u)}{T})$ many noisy nodes (p-sums), thus the phase two privacy loss is bounded by

$$\Pr[\mathsf{Lkg} = \mathbf{o}]/\Pr[\mathsf{Lkg}' = \mathbf{o}] \le e^{\log_2(\frac{t-u}{T})\times\epsilon}$$

In addition, when $t-u > L$, then the mechanism incurs privacy loss for both phase one and phase two distortion. By Theorem 14 the total phase one privacy loss is bounded by $\lfloor\frac{t-u}{L}\rfloor \times \epsilon$. For phase two, the max privacy loss is bounded by $e^{\log_2(\frac{L}{T})\times\epsilon}$. As such, we may conclude

$$\Pr[\mathsf{Lkg} = \mathbf{o}]/\Pr[\mathsf{Lkg}' = \mathbf{o}] \le e^{\left(\lfloor\frac{t-u}{L}\rfloor + \log_2(\frac{t-u}{T})\right)\times\epsilon}$$

As summary, the privacy decay function to be

$$\omega(t-u) \le \begin{cases} \lfloor\frac{t-u}{L}\rfloor + \log_2(\frac{L}{T}), & \text{if } (t-u) > L \\ \log_2(\frac{t-u}{T}), & \text{otherwise} \end{cases} \tag{19}$$

Proof of Theorem 16 For the purpose of completeness, we start with the necessary Lemmas which were discussed in [DR+14,WBNM21] but with necessary variations.

Lemma 26. *Given n independent and identically distributed (i.i.d.) Laplace random variables $X_1, X_2, ..., X_n$ drawn from* $\mathrm{Lap}(\frac{\Delta}{\epsilon})$. *Let $X = \sum_{i=1}^n X_i$, $0 < \alpha \le n\frac{\Delta}{\epsilon}$, then:*

$$\Pr[\,X \ge \alpha] \le e^{\left(\frac{-\alpha^2\Delta^2}{4n\epsilon^2}\right)}$$

Proof. Please refer to the proof to Lemma 12.2 in [DR+14] or proof to Theorem 6 in [WBNM21]

Lemma 27. *Given n i.i.d. Laplace random variables, $X_1, X_2, ..., X_n$, where each $X_i \sim$* $\mathrm{Lap}(\frac{\Delta}{\epsilon})$. *Let $X = \sum_{i=1}^n X_i$, and $\beta \in (0,1)$, the following inequality holds*

$$\Pr\left[X \ge 2\frac{\Delta}{\epsilon}\sqrt{n\log\frac{1}{\beta}}\right] \le \beta$$

Proof. As per Lemma 26, let $e^{(\frac{-\alpha^2\Delta^2}{4n\epsilon^2})} = \beta$, and take log on both sides, one can obtain the following:

$$\begin{aligned} &\log\left(e^{(\frac{-\alpha^2\Delta^2}{4n\epsilon^2})}\right) = \log(\beta) \Rightarrow \log\left(e^{(\frac{\alpha^2\Delta^2}{4n\epsilon^2})}\right) = \log\left(\frac{1}{\beta}\right) \\ \Rightarrow\ &\frac{\alpha^2\Delta^2}{4n\epsilon^2} = \log\left(\frac{1}{\beta}\right) \Rightarrow \alpha^2 = \frac{4n\epsilon^2}{\Delta^2}\log\left(\frac{1}{\beta}\right) \\ \Rightarrow\ &\alpha = 2\frac{\Delta}{\epsilon}\sqrt{n\log\frac{1}{\beta}} \end{aligned} \tag{20}$$

Proof. (**Theorem** 16) When adopting stake distortion, the inference errors consists of two parts: (i) the error due to injected DP noises and (ii) the error caused by the inference algorithm. By Theorem 28, the part (ii) noise is bounded by η. By Lemma 27, when $\mathcal{F}_{\mathsf{Timer}}$ and $\mathcal{F}_{\mathsf{Bin}}$ are applied, the part (i) errors are bounded by $O(\frac{\alpha}{\epsilon})$, and $O(\frac{\sqrt{L}\alpha}{\epsilon})$, respectively.

Proof of Theorem 17

Proof. (**Theorem 17**) Let $X_a \le \frac{1-\gamma}{3}S$, $X_h \ge \frac{2+\gamma}{3}S$ to be stake controlled by the adversary and the honest parties, respectively. We denote the distorted stake as $\tilde{X}_a \leftarrow X_a + Y^{n_a}$, and $\tilde{X}_h \leftarrow X_h + Y^{n_h}$, respectively for adversary and honest, where Y^x is the summation of x *i.i.d.* Laplace random variables, and n_a, n_b denotes the number of honest and malicious parties, respectively, such that $n_a + n_b = n$. To ensure that the corrupted stake after distortion is bounded by $\frac{1}{3}$, we need to have $\tilde{X}_h - 2\tilde{X}_a > 0$ with high probability. Knowing that:

$$\tilde{X}_h - 2\tilde{X}_a = (X_h - 2X_a) + Y^{n_h} - 2Y^{n_a} = \gamma S + Y^{n_h} - 2Y^{n_a} \tag{21}$$

Since the random variables are symmetric, without loss of generality, we may assume the following

$$\Pr\left[\tilde{X}_h - 2\tilde{X}_a < 0\right] = \Pr\left[Y^{n_h} + 2Y^{n_a} > \gamma S\right] \tag{22}$$

By Lemma 12.2 in [DR+14], we can obtain that for $\beta \in (0,1)$

$$\Pr\left[Y^{n_h} + 2Y^{n_a} > \sqrt{\log\frac{2}{\beta}}\frac{\alpha\left(\sqrt{n_h} + 2\sqrt{n_a}\right)}{\epsilon S}\right] \le \beta \tag{23}$$

By setting $\gamma \ge \sqrt{\log\frac{2}{\beta}}\frac{\alpha(\sqrt{n_h}+2\sqrt{n_a})}{\epsilon S}$, with probability at least $1-\beta$, the adversary controlled stake after distortion does not exceed $\frac{1}{3}$ of the total fraction.

In what follows we show that $\sqrt{n_h} + 2\sqrt{n_a} < 1.74\sqrt{\frac{S}{v}}$. Let $f(x,y) = \sqrt{x} + 2\sqrt{y}$ such that $x > 0$ and $y > 0$, then

$$\frac{\partial f(x,y)}{\partial x} = \frac{1}{2\sqrt{x}}, \quad \frac{\partial f(x,y)}{\partial y} = \frac{1}{\sqrt{y}} \tag{24}$$

By Equation 24, we conclude that $f(x,y)$ is a monotonically increasing function related to x and y. Therefore, $\max f(x,y) = f(\max(n_h), \max(n_a))$. Given minimum staking amount v, $\max(n_h) = \frac{hS}{v}$, and $\max(n_a) = \frac{(1-h)S}{v}$, where $h > \frac{2}{3}$ denotes the fraction of honest stake. Obviously, the value $\sqrt{h} + 2\sqrt{1-h}$ decreases when h increases, thus the max value is $\sqrt{(\frac{2}{3})} + 2\sqrt{\frac{1}{3}} < 1.74$.

Thus, with knowledge of S and v for any system, one may derive an upper bound on slack γ by replacing $\sqrt{n_h} + 2\sqrt{n_a}$ with $1.74\sqrt{\frac{S}{v}}$, this proves the theorem.

D Stake Inference Attack Continued

D.1 Binary search with backtracking.

The algorithm is given the total stake S, a target P_v with stake f_v, $\theta, \tau > 0$, and a balanced (stake) binary tree ST with each node labeled with a stake segment $[f_a, f_b] \subseteq [0, f]$. Specifically, the root is labeled with $[0, f]$, and for every internal node v with label $[f_a, f_b]$, its left and right child is labeled with $[f_a, m]$ and $[m, f_b]$, respectively, where $m = \lfloor\frac{f_a+f_b}{2}\rfloor$, and all leaf nodes satisfy $f_b - f_a = \frac{\theta}{S}$.

The algorithm proceeds with a random walk in ST starting from the root, and for each round t (assuming at node $v(t)$), it performs the following. First, it runs two "one-flip-test (OFT)" to compare f_v with f_a and f_b (labels of $v(t)$). To invoke an OFT between any value f_x and f_v, the attacker creates two conflicting transactions (i.e., as long as one is confirmed, the other one is invalid), tx_0, and tx_1, broadcast them to everyone but (i) delays tx_0 to P_v and (ii) removes tx_1 from the corrupted users with stake f_x. If tx_0 (*resp.* tx_1) is not confirmed in the next time slot, the test claims $f_v > f_x$ (*resp.* $f_v < f_x$), otherwise the test claims a random comparison result. This test design allows us to flip and compare two simulated coins at the same time. If $f_v < f_a$ or $f_v > f_b$, the algorithm moves back to the parent node of $v(t)$, otherwise evaluate another "one-flip-test" with $m \leftarrow \frac{f_a+f_b}{2}$, and moves to the left child if $f_v < m$, and right otherwise.

When a leaf node is reached, the algorithm runs an exit test by tossing the coins corresponding to f_a, f_v, and f_b, $(\frac{\log 1/\delta}{\tau^2})$ many times each to obtain the estimated heads probabilities $\hat{p}_a, \hat{p}_v$, and $\hat{p}_b$, respectively. If $\hat{p}_v \pm \tau$ intersects $[\hat{p}_a, \hat{p}_b]$, it computes $q^{-1}(p_a, p_b)$ and return $\frac{f_a+f_b}{2}$, otherwise moves back to the parent node.

Theorem 28. *With probability at least* $1-\delta$*, the algorithm outputs* $\hat{f}_v$ *that* $|\hat{f}_v - f_v| \le \frac{\theta}{S} + \eta$ *with expected* $\Omega(\frac{\log\frac{1}{\delta}\log(\frac{fS}{\theta})}{\tau^2})$ *many (sim) coin flips, where* $\eta = \max(|q^{-1}(p_a-\tau)-f_a|, |q^{-1}(p_b+\tau)-f_b|)$.

We provide formal proofs of Theorem 28 as follows.

Lemma 29. *When the algorithm terminates and outputs $\hat{f}_v$, then with probability at least $1-\delta$, $|\hat{f}_v - f_v| \le \theta + \eta$, where $\eta \leftarrow \max(|q^{-1}(p_a - \tau) - f_a|, |q^{-1}(p_b + \tau) - f_b|)$.*

Proof. *We say that the proof of Lemma 29 is equivalent to show p_a and p_b intersects $p_v \pm \tau$ at the algorithm termination. For any two biased coins with heads probability p and p' such that $|p - q| \ge \tau$, and let $\bar{p}$, $\bar{q}$ denotes the heads frequencies for each coin after $2\frac{1}{\tau^2}\log\frac{2}{\delta}$ many tosses. Note that by Hoeffding's Inequality [HR70],*

$$\Pr[|\bar{p} - p| \ge \frac{\tau}{2}] \le e^{-2\tau^2 \times \frac{\tau^2}{4} \times \log 2 \frac{2}{\delta}} = \frac{\delta}{2}$$

The same holds for q and $\bar{q}$, thus by applying union bound, we say that after $2\frac{1}{\tau^2}\log\frac{2}{\delta}$ many coin tosses on each coin, with probability at least $1 - \delta$, one can correctly distinguish between coin p and q. Therefore, we say that when RdBin *terminates and outputs $\hat{f}_v$ at least one of the following is true with probability at least $1-\delta$:*

1. *$|p_a - p_v| \le \tau$ or $|p_b - p_v| \le \tau$*
2. *$|p_a - p_v| \ge \tau$ and $|p_b - p_v| \ge \tau$ and $p_a \le p_v \le p_b$*

And thus proof completes.

For any leaf node of the binary stake tree, we say a node is a good node if the corresponding endpoints p_a, and p_b intersect with p_v, we denote the set of good (leaf) nodes in the stake tree as $\mathbb{G}$.

Lemma 30. *Let $d(v, \mathbb{G})$ denotes the distance between a true node v and the closest good node in $\mathbb{G}$. Conditional on $v(t) \notin \mathbb{G}$, the probability of $d(v(t), \mathbb{G}) = d(v(t-1), \mathbb{G}) - 1$ is at least $\frac{1}{2} + \tau$.*

Proof. Given the condition that $v(t) \notin \mathbb{G}$, and let p_a, p_b to be $v(t)$'s endpoints, then one of the following is true

1. $p_b < p_v - \tau$ or $p_a > p_v + \tau$
2. $p_a \le p_v \le p$ and $|p_b - p_v| + |p_a - p_v| \ge 2\tau$

Note that given two biased coins p, and q such that $|p - q| \ge \tau$ and $p > q$, the probability that the one flip test outputs the correct comparison is at least $\frac{1}{2} + \tau$. Thus for case one, conditional on $v(t) \notin \mathbb{G}$ the probability of $d(v(t), \mathbb{G}) = d(v(t-1), \mathbb{G}) - 1$ is equivalent to the probability that the OFT outputs the correct claim that $p_b < p_v$ or $p_a > p_v$, which is at least $\frac{1}{2} + \tau$. The same probability can be obtained for case two as well.

Lemma 31. *The algorithm halts and output a result after expected $\Omega(\frac{\log\frac{1}{\delta}\log(\frac{fS}{\theta})}{\tau^2})$ many (sim) coin flips.*

Proof. By Lemma 30 for every step the random walk moving to the correct direction with probability at least $\frac{1}{2}$. Thus by Theorem 3.5 in [FRPU94], the expected steps of the random walk before halting is bounded by $\Omega(\log\frac{fS}{\theta})$, and thus the expected complexity of the entire inference approach is bounded by $\Omega(\frac{\log\frac{1}{\delta}\log(\frac{fS}{\theta})}{\tau^2})$.

D.2 Bayes learner based construction.

Follows our general paradigm, we continue our discussion on another Bayesian learner based SIA construction (BLSIA). The details are provided in Algorithm 3.

Similarly, BLSIA is given $\theta > 0$ and $\tau \in (0, 1)$, the algorithm first evenly divides the attacker's stake f into $m \leftarrow \frac{S}{\theta}$ equal size range segments $\{a_1, a_2, ..., a_m\}$, such that a_i denotes the range $[(i-1)\frac{\theta}{S}, i\frac{\theta}{S}]$. Then, the it proceeds as follows. First, BLSIA assigns an initial weight $w_i \leftarrow \frac{1}{m}$ to each segment a_i, and starts updating them. For each step, BLSIA picks the larges index j such that $\sum_{i=1}^{j} w_j < \frac{1}{2}$, and chooses the middle value indicated by segment a_j to launch a one flip test (Alg 3:4,5). BLSIA then updates each weight accordingly based on the OFT output (Alg 3:6-9). We stress that the weight update process is the direct application of the Bayes posterior probability update process, and in order to ensure the updated weight sum up to 1, all weights are normalized after updates. This process is repeated until it finds a weight w_i such that $w_i > \frac{1}{2}$. The attack then terminates and outputs $\frac{(2i-1)\theta}{2S}$, the middle value indicated by the segment a_i, (Alg 3:10,11) as the inference result.

Theorem 32. BLSIA*'s running time is bounded by expected $O(\frac{\log m}{\tau^2})$ many simulated coin flips.*

To prove this theorem, we first summarizes important notations used in our proof:

– $H(X)$ denotes the entropy of random variable X and $I(X) \leftarrow 1 - H(X)$ is the corresponding negentropy;

Algorithm 3 Stake Inference Attack - II (Bayes learner)

```
    Input: $\theta > 0, \tau \in (0,1)$; $P_v$ with stake $f_v$.
    Output: The approximated stake $\tilde{f}_v$ for party $P_v$.
1:  $m \leftarrow \frac{S}{\theta}$, $\{a_1, a_2, ..., a_m\} : a_i \leftarrow [(i-1)\theta, i\theta]$
2:  Initialize weights $a_i$, $\{w_1, w_2, ..., w_m\} : w_i \leftarrow \frac{1}{m}$
3:  $p \leftarrow \frac{1}{2} + \tau$
4:  for $t = 1, 2, 3...$ do
5:      Select $j$, that $j = \max_k : \sum_{i=1}^{k} w_i < \frac{1}{2}$
6:      Run OFT between $f' \leftarrow \frac{(2j-1)\theta}{2S}$, and $f_v$
7:      if $f' > f_v$ by OFT then
8:          $w_i = \begin{cases} \frac{(1-p)w_i}{(1-p)\sum_{i=1}^{j} w_i + p\sum_{i=j+1}^{m} w_i} & \forall\, i \le j, \\ \frac{p w_i}{(1-p)\sum_{i=1}^{j} w_i + p\sum_{i=j+1}^{m} w_i} & \forall\, j < i \le m. \end{cases}$
9:      else
10:         $w_i = \begin{cases} \frac{p w_i}{p\sum_{i=1}^{j} w_i + (1-p)\sum_{i=j+1}^{m} w_i} & \forall\, i \le j, \\ \frac{(1-p)w_i}{p\sum_{i=1}^{j} w_i + (1-p)\sum_{i=j+1}^{m} w_i} & \forall\, j < i \le m. \end{cases}$
11:     if $\exists_i : w_i > \frac{1}{2}$ then
12:         return $\hat{f}_v \leftarrow \frac{(2i-1)\theta}{2S}$
```

– $H(X)$ denotes the entropy of random variable X and $I(X) \leftarrow 1 - H(X)$ is the corresponding negentropy;

Lemma 33. *If algorithm 3 terminates at round t and outputs a value, then* $H(\mathbf{w}^{(t)}) < \log(3)$.

Proof. If algorithm 3 terminates, then there exists i such that $w_i > \frac{1}{3}$. By the definition of entropy, we compute

$$\begin{aligned} H(\mathbf{w}^{(t)}) &= \sum_{i=1}^{m} -w_i^{(t)} \log(w_i^{(t)}) < \sum_{i=1}^{m} -w_i^{(t)} \log(\frac{1}{3}) \\ &= \log(3) \sum_{i=1}^{m} w_i^{(t)} = \log(3) \end{aligned} \tag{25}$$

Lemma 34. *Given the weights before and after applying the* t^{th} *update as* $\mathbf{w}^{(t-1)}$*, and* $\mathbf{w}^{(t)}$*. The expected information gain after* t^{th} *round of weight update satisfies*

$$\mathbb{E}\left(I(\mathbf{w}^{(t)}) - I(\mathbf{w}^{(t-1)})\right) \ge \frac{2}{9} I(p)$$

Proof. Let $\kappa \leftarrow \sum_{i=1}^{j} w_i^{(t-1)}$, $\mathsf{nor}_0 \leftarrow \frac{1}{p\kappa + (1-p)(1-\kappa)}$, and assume the reverse tagging attack outputs $b = 0$ at t^{th} round. We computes the following

$$\begin{aligned} &I(\mathbf{w}^{(t)} | b = 0) \\ &= \log m + \sum_{i=1}^{j} \mathsf{nor}_0 \cdot p w_i^{(t-1)} \log(\mathsf{nor}_0 \cdot p w_i^{(t-1)}) + \sum_{i=j+1}^{m} \mathsf{nor}_0 \cdot (1-p) w_i^{(t-1)} \log(\mathsf{nor}_0 \cdot (1-p) w_i^{(t-1)}) \\ &= \log m + \mathsf{nor}_0 \cdot p \log(\mathsf{nor}_0 \cdot p) \sum_{i=1}^{j} w_i^{(t-1)} + \mathsf{nor}_0 \cdot p \sum_{i=1}^{j} w_i^{(t-1)} \log(w_i^{(t-1)}) \\ &\quad + \mathsf{nor}_0 \cdot (1-p) \log(\mathsf{nor}_0 \cdot (1-p)) \sum_{i=j+1}^{m} w_i^{(t-1)} + \mathsf{nor}_0 \cdot (1-p) \sum_{i=j+1}^{m} w_i^{(t-1)} \log(w_i^{(t-1)}) \\ &= \log m + \mathsf{nor}_0 \cdot p\kappa \log(\mathsf{nor}_0 \cdot p) - \mathsf{nor}_0 \cdot p H(w_1^{(t-1)}, ..., w_j^{(t-1)}) \\ &\quad + \mathsf{nor}_0 \cdot (1-p)(1-\kappa) \log(\mathsf{nor}_0 \cdot (1-p)) - \mathsf{nor}_0 \cdot (1-p) H(w_{j+1}^{(t-1)}, ..., w_m^{(t-1)}) \end{aligned} \tag{26}$$

Note that, with the same technique, one can obtain the negentropy of $\mathbf{w}^{(t)}$ when $b = 1$. Next, we compute the expected negentropy for $\mathbf{w}^{(t)}$ after weight update:

$$\begin{aligned}
&\mathbb{E}(I(\mathbf{w}^{(t)}))\\
=&I(\mathbf{w}^{(t)}|b=0)\times \Pr[b=0] + I(\mathbf{w}^{(t)}|b=1)\times \Pr[b=1]\\
=&I(\mathbf{w}^{(t)}|b=0)\times \frac{1}{\mathsf{nor}_0} + I(\mathbf{w}^{(t)}|b=1)\times\left(1-\frac{1}{\mathsf{nor}_0}\right)\\
=&I(\mathbf{w}^{(t-1)}) + \log n - 1 + (p\kappa + (1-p)(1-\kappa))\log(\mathsf{nor}_0) + (p(1-\kappa)+(1-p)\kappa)\log\left(\frac{\mathsf{nor}_0}{\mathsf{nor}_0-1}\right) - H(p)\\
=&I(\mathbf{w}^{(t-1)}) + \log n - 1 - \frac{1}{\mathsf{nor}_0}\log\left(\frac{1}{\mathsf{nor}_0}\right) + \left(1-\frac{1}{\mathsf{nor}_0}\right)\log\left(1-\frac{1}{\mathsf{nor}_0}\right) - H(p)\\
\geq& I(\mathbf{w}^{(t-1)}) - \frac{1}{\mathsf{nor}_0}\log\left(\frac{1}{\mathsf{nor}_0}\right) + \left(1-\frac{1}{\mathsf{nor}_0}\right)\log\left(1-\frac{1}{\mathsf{nor}_0}\right) - H(p)\\
=&I(\mathbf{w}^{(t-1)}) + H\left(\frac{1}{\mathsf{nor}_0}\right) - H(p)
\end{aligned} \tag{27}$$

According to Eq 26, we can obtain that

$$\mathbb{E}\left(I(\mathbf{w}^{(t)}) - I(\mathbf{w}^{(t-1)})\right) \geq H\left(\frac{1}{\mathsf{nor}_0}\right) - H(p)$$

In what follows we show that the $H\left(\frac{1}{\mathsf{nor}_0}\right) - H(p)$ is at least $\frac{2}{3}I(p)$. Assuming that the range segment partition selects index j such that $\sum_{i=1}^{j} w_i \leq \frac{1}{2}$ and $\sum_{i=1}^{j+} w_i \geq \frac{1}{2}$. In the ideal case, if the partition perfectly divide weights into two groups with summation equals to $\frac{1}{2}$, then $H\left(\frac{1}{\mathsf{nor}_0}\right) = 1$, and the expected information gain equals to $1 - H(p) \geq 2\tau^2 \geq \frac{2}{9}\tau^2$. If it is not a perfect partition, and assume a slack ω such that $\kappa = \sum_{i=1}^{j} w_i \geq \frac{1}{2} - \omega$. Additionally, its obvious that $\omega < \frac{1}{3}$, otherwise the algorithm terminates at $t-1$ round. Now, we compute the following term

$$\begin{aligned}
H\left(\frac{1}{\mathsf{nor}_0}\right) =& H\left(p\kappa + (1-p)(1-\kappa)\right) \geq H\left(\frac{1}{2} + \omega(1-2p)\right)\\
\geq& 1 - (2\omega(1-2p))^2 = 1 - 4\omega^2(1-2p)^2\\
=& 1 - 16\omega^2\tau^2
\end{aligned} \tag{28}$$

Putting it all together we can obtain that

$$\begin{aligned}
\mathbb{E}\left(I(\mathbf{w}^{(t)}) - I(\mathbf{w}^{(t-1)})\right) \geq& 1 - H(p) - 16\omega^2\tau^2\\
\geq& 2\tau^2(1-8\omega^2) \geq \frac{2}{9}\tau^2
\end{aligned} \tag{29}$$

Proof. (**Theorem 32**) According to Lemma 33, when BLSIA terminates at round t, then $H(\mathbf{w}^{(t)}) < \log(3)$. Note that in the initial stage, we assign all weights as $\frac{1}{m}$, therefore $H(\mathbf{w}^{(0)}) = \log(m)$, where $\mathbf{w}^{(0)}$ denotes the initial weight assignments. Thus the expected rounds of simulated coin flips is captured by the following term

$$\frac{\log m - \log 3}{\mathbb{E}\left(I(\mathbf{w}^{(t)}) - I(\mathbf{w}^{(t-1)})\right)} \leq \frac{9\log m}{2\tau^2} \sim O(\frac{\log m}{\tau^2}) \tag{30}$$

Theorem 35. *Given $\theta > 0$ and $\tau \in (0,1)$,* BLSIA *finds the correct stake approximation $\hat{f}_v$, such that $|\hat{f}_v - f_v| \leq \theta + \eta$ with constant failure probability.*

Proof. (**Theorem 35**) Note that the weight update process can also be abstracted as a voting procedure. Each round, the algorithm divides the weights into two parts and votes (by multiplying those weights by a factor of p) for the partition that is considered to include the correct range segment, a_i. Now let's assume the algorithm finds the incorrect segment, say a_j, where a_j does not contain f_v, this means that a_j received at least one vote more than the correct segment a_i. Given that the voting procedure is p-accurate, saying that with probability at least p the algorithm votes for a_i. By Hoeffding's inequality, the probability of such incorrect segment a_j get more votes than the correct segment a_i is bounded by

$$2e^{-2\tau^2 O(\frac{\log m}{\tau^2})}$$

which is at most a constant factor.

E Stake Distortion with Gaussian Noise

As discussed before, one may also choose to follow the *approximated DP* [DR$^+$14] notions by adding an additive slack δ to the right side of Equation 2 in Definition 10. This derives a relaxed definition, which as per Lemma 3.17 in [DR$^+$14], is equivalent to saying that Equation 2 holds with probability $1-\delta$. To distinguish, we refer to the leakage that satisfies the relaxed privacy notion as $(\epsilon, \alpha, \delta)$-private leakage. There are certain benefits for the relaxation, for instance, it provides a tighter privacy composition bound [KOV15]. This property further leads to a smoother privacy decay over time. In what follows, we examine the stake distortion approach that utilizes Gaussian noise, $\mathcal{N}(0, \alpha^2\sigma^2)$, which eventually derives $(\epsilon, \alpha, \delta)$-private leakage but provides a smoother privacy decay.

E.1 Analysis of $\mathcal{F}_{\mathsf{Timer}}$, and $\mathcal{F}_{\mathsf{Bin}}$ mechanisms under Gaussian noises

Theorem 36. *With Gaussian noise $\mathcal{N}(0, \alpha^2\sigma^2)$ such that $\sigma^2 = \frac{2\ln(1.25\delta)}{\epsilon^2}$, $\mathcal{F}_{\mathsf{Timer}}$, and $\mathcal{F}_{\mathsf{Bin}}$ leaks $(\alpha, \epsilon, \delta)$-DP leakage with $\omega(t-u) \le 0.5k\epsilon + \sqrt{k \log \frac{1}{\delta}}$, where:*

$$k = \begin{cases} \lfloor \frac{t-u}{T} \rfloor, & \text{if } \mathcal{F}_{\mathsf{Timer}} \\ \lfloor \frac{t-u}{L} \rfloor + \log(L), & \text{if } \mathcal{F}_{\mathsf{Bin}} \text{ and } (t-u) > L \\ \log(\frac{t-u}{T}), & \text{if } \mathcal{F}_{\mathsf{Bin}} \text{ and } (t-u) \le L \end{cases} \tag{31}$$

Proof. (**Theorem** 36) We consider same notations as used in the proof of Theorem 14. We abstract the stake distortion at time j as $\mathcal{M}_j(s) \leftarrow s + \mathcal{N}(0, \alpha^2\sigma^2)$. As per [DR$^+$14], $\Pr[\mathcal{M}_j(s) = o] \le e^\epsilon \Pr[\mathcal{M}_j(s') = o] + \delta$ when $\sigma^2 = \frac{2\ln(1.25\delta)}{\epsilon^2}$. Similarly, we compute same term as Equation 17 and derive the following inequality.

$$\prod_{j=1}^{t} \frac{\Pr\left[\mathcal{M}_j(s_j) = o_j \mid *\right]}{\Pr\left[\mathcal{M}_j(s'_j) = o_j \mid *\right]} \le \prod_{k=1}^{t-u} \frac{\Pr\left[\mathcal{M}_u(s_u) = o_u \mid *\right]}{\Pr\left[\mathcal{M}_u(s'_u) = o_u \mid *\right]} \tag{32}$$

By the theorem of Concentrated Differential Privacy [DR16], Gaussian mechanism satisfies ρ-zCDP with $\rho = \frac{1}{2\sigma^2}$, and by the composition theorem of ρ-zCDP. Total privacy loss for Gaussian mechanism after k runs satiesfies $(\frac{k}{2\sigma^2})$-zCDP, which is $(\frac{k}{2\sigma^2} + 2\sqrt{\frac{k}{2\sigma^2}\log(\frac{1}{\delta})}, \delta) - DP$. Thus we can obtain

$$\begin{aligned} \omega\epsilon &= \frac{k}{2\sigma^2} + 2\sqrt{\frac{k}{2\sigma^2}\log(\frac{1}{\delta})} \\ &= \frac{k\epsilon^2}{4\ln(1.25\delta)} + \frac{\epsilon}{\sqrt{2}}\sqrt{k\log(\frac{1}{\delta})} \end{aligned} \tag{33}$$

Thus we can obtain $\omega \le \frac{k\epsilon}{2} + \sqrt{k \log \frac{1}{\delta}}$. Knowing that the privacy decay of $\mathcal{F}_{\mathsf{Timer}}$ is subject to $\lfloor \frac{t-u}{T} \rfloor$-fold privacy composition where the privacy decay of $\mathcal{F}_{\mathsf{Bin}}$ is subject to $\log(\frac{t-u}{T})$-fold composition when $(t-u) \le L$ or $(\lfloor \frac{t-u}{L} \rfloor + \log(L))$-fold composition when $(t-u) > L$.

According to Theorem 36 one may replace the noise in $\prod_{\mathsf{LM}^*}$ from Laplace to Gaussian noise for a smoother privacy decay. In what follows, we derive the safety bounds for stake distortion with Gaussian noise, $\mathcal{N}(0, \alpha^2\sigma^2)$.

Theorem 37. *Consider the same setting as Theorem 17, and let $\Gamma = \frac{1.74\alpha\sigma}{S}\sqrt{\frac{S}{v}} \log \frac{1}{\beta}$. Assuming the adversary controls $f = \frac{1-\gamma}{3}$ of the total stake that $\gamma \ge \Gamma$ (resp. $\gamma \ge \sqrt{\log L} \times \Gamma$), then with probability at least least $1-\beta$, $\tilde{f} < \frac{1}{3}$ under $\mathcal{F}_{\mathsf{Timer}}$ (resp. $\mathcal{F}_{\mathsf{Bin}}$).*

Lemma 38. *Let $Y^m = \sum_i^m Y_i$, where each Y_i is an independent Gaussian random variable sampled from the distribution $\mathcal{N}(0, \sigma^2)$.*

Proof. By Markov inequality

$$\begin{aligned} \Pr\left[Y_i \ge \gamma\right] &= \Pr\left[e^{-tY_i} \ge e^{-t\gamma}\right] \le \frac{\mathbb{E}(e^{-tY_i})}{e^{-t\gamma}} = \frac{M_Y(-t)}{e^{-t\gamma}} \\ \Rightarrow \Pr\left[Y^m \ge \gamma\right] &\le \frac{\prod_i^m M_Y(-t)}{e^{-t\gamma}} \\ &\le \min_{t>0} \frac{e^{\frac{m\sigma^2 t^2}{2}}}{e^{t\gamma}} = \min_{t>0} e^{\frac{m\sigma^2 t^2}{2} - \gamma t} \end{aligned} \tag{34}$$

Knowing that the minimum value of $\frac{m\sigma^2 t^2}{2} - \gamma t$ is when $t = \frac{\gamma}{\sigma^2}$, thus $\min_{t>0} \frac{m\sigma^2 t^2}{2} - \gamma t = e^{\frac{-\gamma^2}{2m\sigma^2}}$. Let $\beta = e^{\frac{-\gamma^2}{2m\sigma^2}}$, we obtain $\gamma = \sigma\sqrt{2m \log\frac{1}{\beta}}$, therefore we conclude that

$$\Pr\left[Y^m \geq \sigma\sqrt{2m\log\frac{1}{\beta}}\right] \leq \beta \tag{35}$$

Proof. (**Theorem** 37) By Lemma 38 and Equation 22, we can conclude that by setting $\gamma \geq \Gamma$ (resp. $\gamma \geq \log L\Gamma$), with probability at least $1-\beta$, the distorted stake satisfies $\tilde{X}_h - 2\tilde{X}_a > 0$ under $\mathcal{F}_{\mathsf{Timer}}$ (resp. $\mathcal{F}_{\mathsf{Bin}}$).

Theorem 39. *Consider the same setting as Theorem 16. By injecting noise $\mathcal{N}(0, \alpha^2\sigma^2)$. Extending $\prod_{\mathsf{LM}}$ to $\prod_{\mathsf{LM}^*}$ by mounting with $\mathcal{F}_{\mathsf{Timer}}$, and $\mathcal{F}_{\mathsf{Bin}}$ yields SIA errors in $\mathsf{Err} + O(\alpha\sigma)$, and $\mathsf{Err} + O(\sqrt{L}\sigma\alpha)$, respectively.*

Proof. (**Theorem** 39) The proof of this theorem is a direct application of Lemma 38.

E.2 Protocols with Gaussian noises

It's not hard to adapt stake distortion protocols, $\prod_{\mathsf{Timer}}$ and $\prod_{\mathsf{Bin}}$, to cope with Gaussian noises. The only change is to transform the uniformly distributed randomness into a Gaussian random variable (was Laplace random variables). This can be done by the following approach. Given two randomness z^0, z^1 which are obtained follow Protocol 6.2:3, then one can transform the randomness to Gaussian noise by computing $z = \sigma\sqrt{-2\ln(z^0)}\cos(2\pi z^1)$ [Box58]. Similarly, one can approximate the complex circuits such as $\cos(x)$, $\sqrt{x}$ with polynomials by using Remez algorithm [Taw05].