Beyond Anonymity Sets: A Security Model for Distributed Shuffling in Adversarial Environments

Adrian Cinal
adrian.cinal@nask.pl
NASK National Research Institute
Warsaw, Poland
Oktawave
Warsaw, Poland

Oliwer Sobolewski
oliwer.sobolewski@nask.pl
NASK National Research Institute
Warsaw, Poland
Oktawave
Warsaw, Poland

Gabriel Wechta
gabriel.wechta@nask.pl
NASK National Research Institute
Warsaw, Poland
Oktawave
Warsaw, Poland

Filip Zagórski*
filip.zagorski@uwr.edu.pl
University of Wroclaw
Wroclaw, Poland
Oktawave
Warsaw, Poland

Abstract

Distributed shuffling is a core primitive underlying mix-nets, electronic voting, and, more recently, single secret leader election (SSLE) protocols for proof-of-stake blockchains. In these settings, a collection of resource-constrained parties jointly permutes a list of ciphertexts or commitments in order to conceal the correspondence between inputs and outputs. Existing security analyses of such protocols typically rely on heuristic anonymity measures or implicitly assume honest behavior; therefore, they fail to capture statistical dependencies that arise when shuffling is partial and some participants are corrupted.

In this work, we introduce a new security model for distributed shuffling that accurately reflects the setting of real-world attacks by explicitly accounting for adversarial corruption and information leakage. Our model allows an adversary to corrupt a subset of shufflers and track selected elements throughout the execution, as well as query the resulting permutation afterwards. We then turn to studying Whisk in this model, the shuffle-based SSLE mechanism proposed for Ethereum. Leveraging Markov-chain techniques, we show that security can only be guaranteed for more stringent parameters than currently proposed. We evaluate the performance impact of changing these parameters on the Ethereum network.

*Corresponding author.

This research was supported by the National Centre for Research and Development under the 'Next Gen Cloud' project (grant No. KPOD.05.10-IW.10-0003/24) funded in the 'Important Project of Common European Interest - Cloud Infrastructure and Services (IPCEI-CIS)' call.

CCS '26, The Hague, Netherlands
© 2026 Copyright held by the owner/author(s).
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS '26), November 15–19, 2026, The Hague, Netherlands*, https://doi.org/10.1145/3830454.3832728.

CCS Concepts

• **Security and privacy → Cryptography**; **Distributed systems security**; *Denial-of-service attacks*; • **Mathematics of computing** → *Markov processes.*

Keywords

Proof of Stake, single secret leader election, Ethereum, DOS attack, anonymity, mixing time, Markov chain

ACM Reference Format:
Adrian Cinal, Oliwer Sobolewski, Gabriel Wechta, and Filip Zagórski. 2026. Beyond Anonymity Sets: A Security Model for Distributed Shuffling in Adversarial Environments. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3830454.3832728

1 Introduction

Distributed shuffling is a fundamental cryptographic primitive with applications spanning mix-nets for anonymous communication [12, 26], electronic voting [37, 39], privacy-preserving cryptocurrencies [6], and, more recently, single secret leader election (SSLE) protocols for Proof-of-Stake blockchains [5]. In distributed shuffling, a set of parties jointly permutes, in rounds, a list of n ciphertexts or cryptographic commitments so as to hide the correspondence between inputs and outputs. Beyond correctness, the central security goal is *anonymity*: even a powerful adversary observing the protocol execution and capable of corrupting some shufflers or tracing some ciphertexts across rounds should be unable to link an honest participant's input to its final position with non-negligible advantage over guessing.

In practical deployments, the above security requirement must be reconciled with computational limitations of the shufflers as well as timing constraints on how long a round of shuffling may take. Indeed, correctness of each round of shuffling must typically be attested to via zero-knowledge arguments [3, 22, 29] that optimize for verifier as opposed to prover speed. When used in, e.g., a cryptocurrency setting [27] where shufflers must make, say, a

12-second slot, shuffling the entire n-element list and providing the corresponding proof of correctness is prohibitively expensive. Therefore, only a small subset of k elements is typically shuffled per round.[1] These constraints have motivated a line of research into how to achieve good shuffling through repeated "local" mixing rather than a single "global" permutation [23, 31, 36]. Existing security analyses of distributed shuffling, however, suffer from a common limitation: they rely on coarse or heuristic anonymity measures that do not fully capture adversarial knowledge and dependencies introduced by partial mixing. In particular, notions such as *anonymity set size*, widely used in the analysis of mix-nets and SSLE protocols, implicitly assume a *uniform distribution* on the set of elements, ignoring correlations between their positions that allow an adversary who learns limited auxiliary information to infer the positions of many more elements [5, 27]. These oversimplifying notions have, perniciously, found their way to technical proposals of real-world deployments [27], and there is little in the literature in the way of formal treatment of distributed shuffling that accounts for adversarial corruption and information leakage. Most notable is the work of Larsen et al. [31], whose formal model and the results derived therein have been used to justify parameter selection for the SSLE protocol Whisk proposed for the Ethereum cryptocurrency [27].

In this work, we revise the model due to Larsen et al. [31], highlighting various insufficiencies of the original and hence undermining many previous conclusions, in particular those that apply to Whisk. While our motivation is rooted in blockchain-based SSLE and related systems, the refined model is generic and applies to a wide class of distributed shuffling mechanisms. It captures (i) a bounded number of corrupted shufflers that may deviate arbitrarily from the protocol, (ii) adversarial tracking of selected elements across rounds, and (iii) realistic constraints on local shuffling capacity. We also improve the analysis of Whisk from [31] by employing standard tools such as path coupling of Markov chains and strong stationary times to get quantitative and composable anonymity guarantees.

1.1 Our Contributions

We make the following contributions in this work:

(1) We point out the failure of the security model for distributed shuffling due to Larsen et al. [31] to capture real-world attacks and, indeed, its limited utility for reasoning about attacks it does capture in theory. In particular, we show that it does not apply to the setting where the protocol proposed by Larsen et al. [31] is most prominently used which is the Ethereum cryptocurrency's [9] SSLE protocol called *Whisk* [27, 28].
(2) We propose a revised model that accounts for the adversary's query access to the resulting permutation and give illustrative separating examples that demonstrate why the previous model is insufficient.
(3) We analyze the number of (honest) rounds needed to get ε-close to a uniformly random permutation with Whisk as defined today [27] as well as its variant originally proposed for Ethereum [30] and conclude there is no theoretical justification for setting the batch size k so small. Indeed, since we point out the analysis due to Larsen et al. [31] is not applicable to $k = 128$, our results are the first to apply to Ethereum. We leverage Markov-chain techniques in both our analyses.
(4) We establish a conceptual link between distributed shuffling and a generalized version of Randomized Partial Checking (RPC) [26, 34] and show the analogy in reasoning about the security of the two schemes.
(5) We evaluate the performance limitations cited by the Ethereum community as the reason for setting the batch size k so small and conclude they are exaggerated. Our benchmarks were run on commodity hardware.

[1] Systematically partitioning the set of ciphertexts into batches also potentially enables a level of parallelism where different shufflers can independently and concurrently perform their mixing rounds.

2 Preliminaries

2.1 Ethereum's Consensus and SSLE

This section elaborates on how distributed shuffling (in an adversarial environment) is related to the problem of consensus in the cryptocurrency Ethereum [9], embedding our work in a real-world context and providing practical motivation for it.

Ethereum relies on a set of *validators* financially incentivized to aggregate transactions into blocks and publish them. To avoid conflicting transaction histories, a *single leader election* mechanism must be present, whereby two nodes do not publish a block at the same time. Unlike in Bitcoin, where the leader is the winner of a race to solve a computational puzzle, in Ethereum, the leader for the next *slot* (approximately 12 seconds) is explicitly elected using public randomness (RANDAO). Currently, the candidates are elected ahead of time for the upcoming 32 slots or one *epoch*. The outcome of this election is public, making Ethereum susceptible to *sequential* (and thus inexpensive) Denial-of-Service (DoS) attacks. If one wanted to sabotage the currency, one could keep targeting the single block proposer elected for each slot [8]. With attributing IP addresses to Ethereum accounts demonstrated to be perfectly feasible in the literature [21, 24], the community has turned to providing leader anonymity at the consensus layer. Specifically, a proposal was put forward to make the block proposer selection process secret [27]. Attackers wanting to disable Ethereum would now have to DoS the majority of, if not all, potential candidates *in parallel*.

This can be viewed as transitioning from merely single leader election to *single secret leader election* (SSLE) [5, 8], where a number of candidates work together to elect from among themselves a single leader in such a way that nobody but this leader will know the result of the election. One particular way of constructing an SSLE protocol is with *distributed shuffling* [31] of the candidate list. Specifically, suppose each candidate registers a *commitment* (called a *tracker* in [27]) of the form (g^u, g^{uv}), for a DDH-hard group $\langle g \rangle$, such that only they know the secret key v. We can then take the list of such trackers, rerandomize and permute it, and then sample at random from this permuted and rerandomized list an element (g^{ur}, g^{uvr}). The owner of the key v and *only* the owner of v will be able to tell that it was their tracker that has been selected. See also [8].

2.2 Whisk the Old and the New

Whisk. The Ethereum research community has proposed Whisk [27, 28] as its distributed shuffling protocol. Whisk works as follows: each day (2^{13} slots) a list of $n = 2^{14}$ candidates is selected from the pool of available validators via the RANDAO randomness beacon. Over the course of the day, the $t = 2^{13} - 32$ block proposers for this day shuffle and rerandomize this list of trackers sequentially by sampling uniformly at random a subset of size k, permuting, and rerandomizing it.[2] This is directly lifted from the distributed shuffling protocol due to [31].

The k permuted trackers are published alongside the blocks that the proposers publish anyway, together with a zero-knowledge argument showing the consistency of the shuffle. Due to efficiency constraints imposed by the use of this zero-knowledge argument, k is chosen to be much smaller than n (currently: $k = \sqrt{n} = 2^7$).[3] After [31]:

> *Two real-world efficiency constraints are important to point out here. Every shuffler needs to speak in a timely manner, yet at the same time, they need to provide zero-knowledge proofs attesting to the correctness of their performed shuffle. These two constraints mean that no shuffler has enough time to permute the full vector at once.*

As we will see, this has implications for the time it takes to shuffle the list thoroughly and indeed whether it has been sufficiently mixed after $t = 2^{13} - 32$ slots.

After the t rounds of shuffling and a cooldown phase of 32 slots comes the next day and RANDAO selects for each slot a new block proposer at random from the permuted candidate list.[4] In their block, the proposer performs a shuffling step for the next day (with a new candidate list), as well as proves in zero knowledge that they are indeed the proposer selected for the current slot.

WhiskRS. In an older version of the Whisk proposal, a different underlying shuffling scheme was used, called a FeistelShuffle [27, 30], which arranged the $n = k^2$ trackers into a $k \times k$ matrix M and employed a systematic sweep across its rows inspired by the square lattice shuffle [23]. We shall refer to this original version of the Whisk protocol as *WhiskRS*. In WhiskRS, each shuffler permutes and rerandomizes one row of the matrix, and k such permutations (called Stirs and potentially computed in parallel by k shufflers[5]) are followed by a public permutation called the Disperse step which relocates the elements according to the following rule:

$$M[x, y] \mapsto M[y, (x + y^3) \bmod k] .$$

Importantly, this map has the property that each element from a given row lands in a different row in the next step.

Because of its structure, in WhiskRS, one must distinguish between a "round" corresponding to a single Stir and a "megaround" corresponding to k Stirs followed by Disperse. Both levels of granularity will be of use to us. On the one hand, each round (single Stir) corresponds to a single shuffler, i.e., a single entity in the blockchain network, which makes round-based analysis natural when considering corruptions (see, e.g., Definition 3.4). Viewing WhiskRS in terms of megarounds (k Stirs and Disperse), on the other hand, lends itself more naturally to Markov-chain analysis (see Section 4).

2.3 Markov Chains

Any shuffling protocol, distributed or not, induces a probability distribution on the symmetric group S_n, the set of all permutations of n elements [1], and the protocol itself can be seen as a random walk in this group. The extent to which anonymity is preserved is then governed by how close the resulting distribution is to the uniform distribution on S_n. Indeed, one can relate (see, e.g., Theorem 5.3) an adversary's advantage in deanonymizing to statistical distance measures such as total variation distance (TVD) or separation distance (both defined below). These are standard tools in cryptography [17, 35]. In fact, security of many cryptographic schemes and protocols, for example, RC4 [35], Randomized Partial Checking [26], or, more recently, ThorPIR [19], has been analyzed using the mixing times of the underlying Markov chains, i.e., the smallest number of steps t it takes for the chain's distribution to be ε-close to its stationary distribution in terms of TVD (see below). The initial weaknesses of RC4 [20], in particular, were discovered because the number of rounds it performed was significantly below the mixing time of the underlying process [35]. This led to a series of research efforts that demonstrated practical methods for breaking RC4 [40].

As we will be using them extensively in this work, we recall some elementary facts about Markov chains and their limiting behaviour.

Definition 2.1. A sequence of random variables $(X_0, X_1, \ldots)$ is called a *Markov chain with state space* $\mathcal{S}$ if for all $x, y \in \mathcal{S}$, all $t \geq 1$, and all events $H_{t-1} = \cap_{s=0}^{t-1}\{X_s = x_s\}$ satisfying $\Pr[H_{t-1} \cap \{X_t = x\}] > 0$, we have

$$\Pr[X_{t+1} = y \mid H_{t-1} \cap \{X_t = x\}] = \Pr[X_{t+1} = y \mid X_t = x] .$$

Definition 2.1 requires that, the *history* H_{t-1} of the sequence before time t be irrelevant and X_{t+1} be determined solely by the transition matrix P and the state at time t. This property is sometimes referred to as *memorylessness* or the *Markov property*.

Definition 2.2. A distribution π on $\mathcal{S}$ is called a *stationary distribution* of the chain if

$$\pi = \pi P ,$$

i.e., it is invariant under the chain dynamics.

An elementary fact about aperiodic and irreducible Markov chains, which all the chains considered in this work will be, is that they have a unique stationary distribution and "converge" to it in a sense introduced below.

A natural measure of distance between two probability distributions μ and ν defined on a finite (countable) space $\mathcal{S}$ is *total variation distance* (TVD):

$$\|\mu - \nu\| := \frac{1}{2}\sum_{x \in \mathcal{S}} |\mu(x) - \nu(x)| = \max_{A \subseteq \mathcal{S}} |\mu(A) - \nu(A)| .$$

[2] The 32-slot (single-epoch) "cooldown phase" at the end of each day is there to prevent the final shufflers from predicting the RANDAO values that will be used after shuffling to sample the validators for the next day [27].

[3] In fact, due to technical limitations of the zero-knowledge argument used, an overhead of 4 elements is needed, so 124 trackers are actually shuffled each round, see [28]. The present work, however, shows that even $k = 128$ is insufficient, so we neglect this detail.

[4] Note how the *shuffling phase* and the *proposal phase* where the block proposers are sampled from the shuffled candidate list are pipelined (with a "bubble" of 32 slots).

[5] If a block proposer for a given slot does not publish a block, their corresponding Stir is taken to be the identity permutation.

An aperiodic and irreducible Markov chain converges to its unique stationary distribution π in total variation, i.e., $\lim_{n\to\infty} \|X_n - \pi\| = 0$.

One can view the TVD metric as describing how different the two probability distributions are *on average*. For security analyses, however, it is more useful to consider how different μ and ν are *in the worst case*. For this purpose, *separation* is often considered instead [2, 33, 34]:

$$s(\mu, \nu) := \max_{x \in \mathcal{S}} \left(1 - \frac{\mu(x)}{\nu(x)}\right).$$

For any μ and ν, we have $\|\mu - \nu\| \leq s(\mu, \nu)$ [2].

For a Markov chain, the probability distributions involved depend on the initial state (or a distribution thereof), so it often makes sense to consider the *maximum* separation at some time t. When treating shuffling as a Markov chain over S_n, however, we may always think of the initial state X_0 as being the identity permutation.

Definition 2.3. Let $(X_0, X_1, \ldots)$ be a Markov chain with stationary distribution π. A *strong stationary time (SST)* T is a randomized stopping time for the chain such that, for all times $t \geq 0$ and all states $x \in \mathcal{S}$,

$$\Pr[X_t = x \mid T = t] = \Pr[X_t = x \mid T \leq t] = \pi(x).$$

At an SST T and beyond, the chain is in its steady state. Naturally, T is a random variable. We have the following bound on how far away the chain is from its stationary distribution at time t.

Lemma 2.4 (Proposition 3.2 in [2]). *If T is a strong stationary time for a chain $(X_0, X_1, \ldots)$ with stationary distribution π, then, for any $t \geq 0$,*

$$s(X_t, \pi) \leq \Pr[T > t].$$

Definition 2.5. A *coupling* of a Markov chain M is a pair process (X_t, Y_t) such that (1) each of X_t and Y_t is a copy of the Markov chain M, and, (2) if $X_t = Y_t$, then $X_{t+1} = Y_{t+1}$.

To prove our main results we will use the Path Coupling Lemma due to Bubley and Dyer [7] as generalized by Czumaj et al. in [16].

Lemma 2.6 ((Delayed) Path Coupling Lemma). *Let M be a Markov chain with a finite state space Ω, and let $d : \Omega \times \Omega \to \mathbf{R}_{\geq 0}$ be a metric (distance function) on Ω. Suppose that for all pairs of adjacent states $x, y \in \Omega$ (i.e., $d(x, y) = 1$), there exists a coupling (X_t, Y_t) of M such that:*

$$\mathbf{E}[d(X_\delta, Y_\delta) \mid X_0 = x, Y_0 = y] \leq (1 - \eta)$$

for some $\eta > 0$. Then, the mixing time $\tau_{\text{mix}}(\varepsilon)$, defined as the time required for the total variation distance between M's distribution at time t and its stationary distribution to be at most ε, satisfies:

$$\tau_{\text{mix}} \leq \frac{\delta}{\eta}\left(\log \operatorname{diam}_d(\Omega) + \log \frac{1}{\epsilon}\right),$$

where $\operatorname{diam}_d(\Omega)$ is the diameter of the state space Ω with respect to the metric d (i.e., the maximum distance between any two states).

Intuitively, Lemma 2.6 says that, when we want to study a coupling where the two chains X_t and Y_t begin in arbitrary states (e.g., where X_0 is the identity and Y_0 uniform on S_n), we may restrict our attention to states that are adjacent in Ω under d.[6] Indeed, we may break up the *path* between X_t and Y_t into segments of distance 1. Since each segment "collapses" in expectation, the whole path is "contracting" and X_t and Y_t move closer together until they finally couple. For $\delta = 1$, we get the classic result due to Bubley and Dyer [7]. Considering $\delta \geq 1$, proposed by Czumaj et al. [16], is useful, however, as Section 4 will show.

3 Distributed Shuffling

Security of Whisk SSLE for Ethereum has only been informally stated in [27, 28], based on the security of the RandomShuffle protocol for distributed shuffling due to Larsen et al. [31]. This informal security claim seems to be that:

> *the anonymity set increases from the status quo of a single validator to at least 8,192 validators.*

While the notion of an "anonymity set" is sometimes a useful simplification [38], one must not forget that it is, at other times, an oversimplification. Pfitzmann and Hansen [38] define it as "the set of all possible subjects," but neglect the fact that not all subjects are *equally likely*. The latter is, however, what is typically implicitly assumed when cryptographers and laymen alike talk about an anonymity set. We will argue below that such a strong claim is unfounded for the result of running Whisk for $t = 2^{13} - 32$ rounds (recall Section 2.2) as is currently proposed [27]. Indeed, we will argue that the results of Larsen et al. [31] are incorrect and our own analysis based on Markov chains finds no justification for their security claim.

To first convince ourselves that an anonymity set is an altogether misguided notion (only at times useful as a mental model), particularly when considering adversaries capable of making queries (or, in fact, even known-plaintext adversaries as opposed to chosen-plaintext), consider the following example.

Example 3.1. Let $\pi \in \mathcal{S}_n$ be generated in the following way by an algorithm R:

(1) select $r \leftarrow \{1, \ldots, n\}$ uniformly at random,
(2) let $\pi[i] = (i + r \bmod n) + 1$, $i = 1, \ldots, n$.

The "anonymity set" is of size n (the highest possible value!) for each element. But an adversary who knows the mixing algorithm R and learns the position of a single element (via even a known-plaintext query) learns the positions of *all* other elements. Note how, in Ethereum, an adversary *is* capable of learning some pairs $(i, \pi(i))$: they learn them as block proposers for the current day reveal themselves. Even once the $2^{13} - 1$ proposers for today have revealed themselves (i.e., the adversary learned the mapping of their trackers under the permutation π), the last one should still be unpredictable. It does not suffice to say that there are still $2^{13} + 1$ trackers in the "anonymity set," if there are correlations between the revealed entries and the ones that remain secret. The main problem with the notion of an anonymity set is thus that it does not capture statistical dependencies.

A more rigorous way to measure anonymity is with TVD as defined in Section 2.3, which captures any statistical biases that an adversary could exploit. It is straightforward to show that TVD

[6] Note that states adjacent according to d do not necessarily need to be adjacent in our Markov chain: there may be no direct (i.e., single-step) transition between the states on our path.

bounds the advantage of any adversary (in particular, a probabilistic polynomial-time one) in distinguishing between two probability distributions and is therefore a standard tool for proving security properties of cryptographic schemes [4, 15, 25]. Continuing with the example, TVD between the uniform distribution on the set of n elements and the distribution generated by the algorithm R is at least $1 - 1/(n-1)!$.

3.1 Old Security Definitions for Distributed Shuffling

We recount the prevailing security definition for distributed shuffling due to Larsen et al. [31] and expose its flaws, within the context of Ethereum and without. We start by discussing some of its features before giving the full definition.

The security argument for the distributed shuffling protocol used for Whisk [31, Theorem 2] considers an "(α, β)-adversary" that statically corrupts *up to* α ciphertexts (Ethereum trackers) and states that the probability this adversary returns a new mapping $(i, \pi(i))$, where the ith ciphertext had not been corrupted, is bounded above by $2/(n-\alpha)$, assuming the adversary uses their entire "corruption budget" and corrupts *exactly* α ciphertexts. The fact that this bound depends only on the available corruption budget α and *not* the actual number of queries made by the adversary is revisited later in this section. For now observe that the resulting claim is still insufficient (even if it were true, which it is not). Indeed, by randomly guessing, any adversary has probability $1/(n-\alpha)$ of finding the image of some fixed i, so Theorem 2 in [31] holding does not preclude the existence of an adversary with non-negligible advantage. If, for example, $\alpha = n-2$, then $\varepsilon = \frac{2}{n-\alpha} = 1$, so the probability of the adversary's winning is ≤ 1, which is vacuous.

Additionally, the security model in [31] does not account for the block proposers' revealing themselves during Ethereum's *block proposal phase* [27], i.e., for the fact that the adversary learns not only α trackers corrupted a priori but also all of the revealed proposers' trackers. (An alternative model was proposed in an early preprint of [31] that accounted for this somewhat but was removed before publication. The alternative model still suffered from the remaining problems recounted in this section, and, importantly, Whisk [27] has only been analyzed in the flawed model from the published version of [31].) Note that the informal anonymity set claim cited above from [27] describes the scenario where the adversary is asked to guess the last block proposer after $n/2-1$ proposers have already revealed themselves today (in addition to the α proposers controlled/tracked by the adversary). Any security model that is to be applied to Ethereum must therefore take this extra information the adversary gets into account. The model we propose in this paper is applicable to and captures a (residual) anonymity guarantee in the case of $n-2$ trackers having been revealed to the adversary (either through static corruption or through post-shuffling queries).

We now present the security model due to Larsen et al. [31] in full. The definitions expect the input set to be a set of ciphertexts of a semantically-secure and key-private public-key encryption scheme (KGen, Enc, Dec), which can be trivially adapted to cryptographic commitments (as in the case of Whisk). Say a distributed protocol Π is a (t, n, k)-*shuffle* if it shuffles a collection of n ciphertexts (resp., commitments) by repeatedly sampling (according to some rule, not necessarily uniformly or even at random) and shuffling k-element subsets at a time for t rounds, rerandomizing the ciphertexts (resp., commitments) each round. Formally, the protocol Π is specified as a sequence of maps $F_i : S_n \times \{0,1\}^* \to S_n$ that, on input a permutation σ_i at time i and random coins r_i, output a permutation at time $i+1$, namely, $\sigma_{i+1} \leftarrow F_i(\sigma_i, r_i)$, with $\sigma_1 = \mathrm{id}$, the identity, and $\sigma_t = \pi$, the output of the protocol. Each F_i uses its random coins to affect only k out of the n ciphertexts. (Note that if all F_is are the same, then an honest protocol execution corresponds to a time-homogeneous Markov chain.)

Say an adversary $\mathcal{A}$ is an (α, β)-adversary against Π if they corrupt up to β rounds of the shuffling protocol and track up to α ciphertexts through the shuffling. More precisely, this means $\mathcal{A}$ knows the keys (resp., commitment trapdoors) to α of the shuffled ciphertexts (resp., commitments) and can select the randomness r_i for β of the total t applications of the maps F_i. This model is consistent with real-world applications such as Ethereum, where consistency of F_i in each round is verifiable, i.e., the network sees that k ciphertexts (resp., commitments) were touched and a zero-knowledge proof attests to correct rerandomization [29]. Therefore, the only freedom $\mathcal{A}$ has is in biasing the random coins r_i.[7] Write $\langle \Pi(\mathbf{c}), \mathcal{A}(\mathbf{c}) \rangle$ for the execution of a shuffling protocol Π with input vector $\mathbf{c}$ in the presence of an adversary $\mathcal{A}$. (Note that, in [31], the exact meaning behind this notation is not elaborated.) The output of this execution is a permutation π and the adversary's output z.

Definition 3.2 (Correctness [31]). A (t, n, k)-shuffle Π is correct in the presence of an adversary $\mathcal{A}$ if:

$$\Pr\left[\pi \neq \perp \;\middle|\; \begin{array}{l} (\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda)\ \forall i \in [n] \\ c_i \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}_i, i)\ \forall i \in [n] \\ \mathbf{c} \leftarrow (c_1, \ldots, c_n) \\ (\pi, z) \leftarrow\!\$\ \langle \Pi(\mathbf{c}), \mathcal{A}(\mathbf{c}) \rangle \end{array}\right] = 1,$$

where $\perp$ is a special symbol describing that π is not a permutation of input ciphertext values and probability is taken over the random coins of all algorithms involved.

Definition 3.3 (Security [31]). Let $\mathcal{A}$ be a probabilistic polynomial-time (α, β)-adversary. Then, a (t, n, k)-shuffle Π is (ε, δ)-secure in the presence of $\mathcal{A}$, if for all $\mathcal{I} \subset [n] : |\mathcal{I}| \leq \alpha$, it holds with probability at least $1 - \delta$ that:

$$\Pr\left[\pi(i) = j \wedge i \notin \mathcal{I} \;\middle|\; \begin{array}{l} (\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda)\ \forall i \in [n] \\ c_i \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}_i, i)\ \forall i \in [n] \\ \mathbf{c} \leftarrow (c_1, \ldots, c_n) \\ \mathbf{sk} \leftarrow \{\mathsf{sk}_i \mid i \in \mathcal{I}\} \\ \big(\pi, (i, j)\big) \leftarrow \langle \Pi(\mathbf{c}), \mathcal{A}(\mathbf{c}, \mathbf{sk}) \rangle \end{array}\right] \leq \epsilon,$$

where probability is taken over the random coins of all algorithms involved.

We argue that Definition 3.3 uses too many free parameters. In particular, in practice, we are not interested in schemes with non-negligible δ value. Indeed, δ describes the probability that

[7] Note how this formalism can naturally be used to also capture *leakage* as opposed to active round corruption, where the r_is are sampled uniformly but $\mathcal{A}$ *learns* β of them [33].

either (1) the protocol Π succeeded in producing a reasonably good permutation π or that (2) our analysis happens to be correct. No matter the interpretation, Definition 3.3 tells us nothing about the security of Π on the off chance that the event (1 or 2) associated with δ happens. In other words, if we can only say that the probability of $\mathcal{A}$ winning is less than or equal ε with probability $1-\delta$, then with probability δ, $\mathcal{A}$'s advantage is something else entirely that is not accounted for at all in the definition. We should therefore assume the worst and say that, with probability δ, $\mathcal{A}$ always wins (e.g., because Π failed, see interpretation 1 above). The *total probability* that $\mathcal{A}$ wins therefore depends on δ and is equal:

$$\Pr[\mathcal{A} \text{ wins}] \leq \varepsilon \cdot (1-\delta) + \delta\,.$$

If δ is non-negligible, then the security claim is vacuous.

Additionally, as we stated before, the security model only accounts for the α ciphertexts/commitments corrupted by the adversary before the protocol execution, whereas, in Whisk, in the worst case of the last block proposer for the day, the adversary also learns $n/2-1$ additional mappings (under π) *after* the protocol has been executed, and only then makes their guess. It is important to note that this case has been taken into account in the early preprint of [31], where an oracle similar to ours has been proposed, but this enhanced definition had regrettably been removed prior to publication (to the overall detriment of the paper).

One last flaw of the prevailing model is that ε is to be computed as a function of α and β, which are *upper bounds* on the number of corruptions $\mathcal{A}$ chooses to make (they represent $\mathcal{A}$'s "budget" for corruptions). But this budget need not be used by $\mathcal{A}$ in which case the task they are given becomes easier and $\mathcal{A}$ may have non-trivial advantage over guessing without violating the security definition (in other words, Definition 3.3 does not rule out the existence of $\mathcal{A}$). Specifically, consider the security proof Larsen et al. give for Whisk [31, Theorem 2]. They explicitly only consider the case where the number of corrupted indices $|\mathcal{I}| = \alpha$, the full corruption budget, and conclude that:

> *[Whisk] is a (ε, δ)-secure (T, n, k)-shuffle in the presence of a (α, β)-adversary, where $\varepsilon = \frac{2}{n-\alpha} + \mathsf{negl}(\lambda)$.*

However, since when defining an (α, β)-adversary we consider only an *upper bound* on the number of corruptions, any (α, β)-adversary can also be viewed as an $(\alpha+1, \beta)$-adversary that does not use up its whole available corruption budget. Suppose $\mathcal{A}$ is the sole efficient adversary that breaks some shuffling protocol Π with probability p by making $q \leq \alpha$ queries.[8] Suppose $\varepsilon(\alpha) < p < \varepsilon(\alpha+1)$, where $\varepsilon(\cdot)$ is the probability bound considered in Definition 3.3. (For example, for Whisk, Larsen et al. [31] consider $\varepsilon(\alpha) = 2/(n-\alpha)$.) This same adversary, making q queries every time, is an (α, β)-adversary that breaks the (ε, δ)-security of Π (because $p > \varepsilon(\alpha)$) but is an $(\alpha+1, \beta)$-adversary that *does not* break the (ε, δ)-security of Π. By allowing the adversary more power, we made the primitive more secure![9] This lack of monotonicity in α not only contradicts reasonable intuition, but most importantly makes Definition 3.3 operationally unusable. While Larsen et al. [31] remark at one point that the best value of ε we can hope for is $1/(n-|\mathcal{I}|)$ as opposed to $1/(n-\alpha)$, which is true, they nevertheless fall victim to their own modeling and, in the proof, identify $|\mathcal{I}| = \alpha$.

[8] One may imagine that making more queries would lower the probability of success by, e.g., shrinking the set of allowed guesses.

[9] Another way of looking at it is that whether the scheme is "secure" or not against a particular adversary depends on how we choose to frame this adversary.

3.2 New Security Definition for Distributed Shuffling

We therefore propose a new security definition for the distributed shuffling scheme, with clearer security goals, less parametrization, and access to an additional oracle. We assume an adaptive *chosen-ciphertext attack*, where the adversary can choose which mappings (or reverse mappings!) they learn in the query phase.

As in the original model [31], α is a parameter that describes the *corrupted ciphertexts* budget, i.e., the maximum number of elements that the adversary can always track throughout the shuffling, due to having learned their corresponding secret keys (or, in the case of commitments, due to having learned the corresponding openings). We put a bound $\alpha \leq n-2$ since any value higher than $n-2$ makes the experiment trivial (the adversary learns the entire permutation through corruption alone). The second parameter, β, describes the *corrupted shufflers* budget, i.e., the maximum number of shufflers (or shuffling rounds) controlled by the adversary. One new parameter appearing in the definition is γ, which denotes the *query budget*, i.e., the maximum number of queries to the oracles O_π and $O_{\pi^{-1}}$ that the adversary can make.

For simplicity, in the main body of this work we do not make γ a parameter of the adversary and instead allow the adversary the maximum number of queries that do not make the experiment trivial. Specifically, we set $\gamma = n - \alpha - 2$. That is, if the adversary statically corrupts α ciphertexts and then queries the π oracle γ times, they are still left with two ciphertexts to distinguish. A more general model, where γ is a parameter is presented for completeness in Appendix B.

Definition 3.4 (Security [this work]). Let $\mathcal{A}$ be a probabilistic polynomial-time (α, β)-adversary. Then, a (t, n, k)-shuffle Π is (ε, λ)-secure in the presence of $\mathcal{A}$, if for all $C \subset [n] : |C| \leq \alpha \leq n-2$ and $\gamma := n - \alpha - 2$:

$$\left| \frac{1}{n - |C \cup Q|} - p \right| \leq \varepsilon + \mathsf{negl}(\lambda)\,,$$

where

$$p = \Pr\left[\pi(i) = j \;\middle|\; \begin{array}{l} (\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)\ \forall i \in [n] \\ c_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}_i, i)\ \forall i \in [n] \\ \mathbf{c} \leftarrow (c_1, \ldots, c_n) \\ \mathbf{sk} \leftarrow \{\mathsf{sk}_i \mid i \in C\} \\ (\pi, z) \leftarrow\!\!\$\ \langle \Pi(\mathbf{c}), \mathcal{A}(\mathbf{c}, \mathbf{sk}) \rangle \\ Q \leftarrow \emptyset \\ (i, j) \leftarrow\!\!\$\ \mathcal{A}^{O_\pi, O_{\pi^{-1}}}(z) \\ i \notin C \cup Q \end{array}\right]$$

and the probability is taken over the random coins of all algorithms involved. The oracles $O_\pi, O_{\pi^{-1}}$ are defined as follows.

$O_\pi(i)$:

1: **if** $|Q| = \gamma$ **then**
2: **return** $\perp$
3: $j \leftarrow \pi(i)$
4: $Q \leftarrow Q \cup \{i\}$
5: **return** j

$O_{\pi^{-1}}(j)$:

1: **if** $|Q| = \gamma$ **then**
2: **return** $\perp$
3: $i \leftarrow \pi^{-1}(j)$
4: $Q \leftarrow Q \cup \{i\}$
5: **return** i

Intuitively, the new security definition requires the adversary not have any advantage in finding a new mapping under π of an index i not corrupted statically ($i \notin C$) and not queried ($i \notin Q$). Importantly, however, the "difficulty" of the adversary's task "scales" with how many queries or corruptions they have made: the more mapping they have learned, the smaller the remaining "anonymity set" and the target probability of success changes to account for that. One may view this experiment as a computational analogue of requiring that the conditional probability distribution conditioned on $\mathcal{A}$ having made queries Q and corruptions C remains uniform on the remaining $n - |C \cup Q|$ ciphertexts.

Note how, for Whisk and Ethereum, we are interested in shuffling schemes with $\gamma = n/2 - 1$, corresponding to the block proposers revealing themselves throughout the day, and hence $\alpha \leq n/2-1$. No security claims beyond that make any sense. (See also Appendix B for treatment of security with γ being a parameter of the adversary and not fixed as in Definition 3.4.) Finally, our model has a "chosen-ciphertext attack" flavour to it, where the adversary can adaptively query both forward and backward mappings under π. Our model is therefore more general than required for Ethereum. Indeed, for Ethereum it would suffice to consider a "known-plaintext attack" model, where the adversary only learns which candidates were shuffled into some particular m positions (selected by RANDAO) under π for some $m < n/2$. Specifically, in Ethereum, the adversary learns (in the worst case) $\pi^{-1}(r_1), \ldots, \pi^{-1}(r_m)$, where $r_1, \ldots, r_m$ are (public) values obtained from RANDAO for the purpose of proposer selection. Indeed, when the adversary wants to launch a DoS attack, up to m block proposers have already revealed themselves in the current block proposal phase.

4 Analysis of WhiskRS

We start our analysis of Ethereum's SSLE with the older proposal, namely, WhiskRS. Recall from Section 2.2 that WhiskRS arranges the n trackers in a $k \times k$ matrix M and permutes them via k (parallel) private random shuffles of the row (the Stir operations) followed by a public permutation Disperse. We will later argue that this systematic sweep down the rows of M admits a strategy for an (α, β)-adversary to corrupt shufflers adaptively each round and "follow" an element through the permutation, always targeting the row in which this element lands after Disperse. For now, however, consider an adversary incapable of shuffler corruptions, i.e., with $\beta = 0$. We shall use a Markov-chain approach to find the mixing time (the "rate of convergence") of WhiskRS. As we will see, the same approach can be used with Whisk without worrying about $\beta > 0$.

4.1 WhiskRS as a Random Walk

Recall from Section 2.3 that a shuffling protocol is best viewed as a random walk in the symmetric group S_n where we may without loss of generality associate the starting state with the identity permutation.

Let the state at time t be:

$$X_t = \begin{pmatrix} x^t_{1,1} & \cdots & x^t_{1,k} \\ \vdots & \ddots & \vdots \\ x^t_{k,1} & \cdots & x^t_{k,k} \end{pmatrix}$$

Transition to state X_{t+1} proceeds as follows:

(1) to each row $i = 1, \ldots, k$ a Stir operation is applied, namely, a permutation $\pi_i^{(t)}$ is sampled uniformly at random from S_k and applied to the elements in the row,
(2) then the public Disperse permutation is applied,

that is

$$X_{t+1} \leftarrow \text{Disperse}\begin{pmatrix} \text{Stir}(x^t_{1,1}, \ldots, x^t_{1,k}) \\ \vdots \\ \text{Stir}(x^t_{k,1}, \ldots, x^t_{k,k}) \end{pmatrix}.$$

Importantly, note how in the formulation, as well as in the results that follow in this section, we identify time with the number of "megarounds" (recall Section 2.2) comprising k Stirs and a Disperse. This is natural since Stirs are performed in parallel and yields a time-homogeneous Markov chain.

4.2 Coupling

We now prove the upper bound on the mixing time of WhiskRS using a (delayed) path coupling argument (Lemma 2.6).

Theorem 4.1 (Mixing Time of WhiskRS). *Mixing time of WhiskRS for fixed ε is*

$$\tau_{\text{mix}} = O(\sqrt{n} \log n).$$

As per the discussion above, τ_{mix} is expressed in the number of megarounds that "touch" every row of the trackers matrix M. There are k as many shufflers involved (assuming no shuffler speaks more than once), k of them working in parallel at any time.

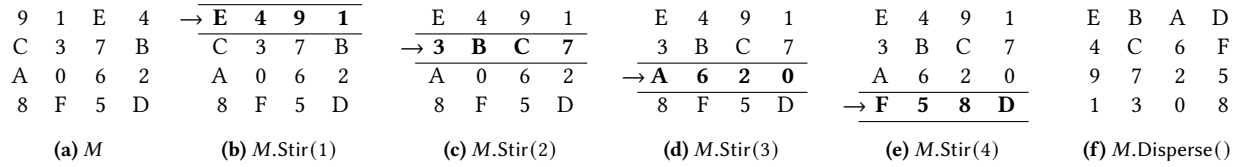
Proof. To apply the Path Coupling Lemma (Lemma 2.6), we need to consider a coupling (X_t, Y_t) for adjacent states where the metric d we use is the minimum number of transpositions needed to go from one state to the next in $\Omega = S_n$. That is, for $\sigma, \sigma' \in S_n$, we say $\sigma \sim \sigma'$ if there exists a transposition (permutation of order 2) $\tau \in S_n$ such that $\sigma' = \sigma\tau$.

Define the coupling as follows. Let X_t progress according to the WhiskRS description above and let Y_t evolve in such a way as to "try" to follow X_t whenever possible. Specifically, let

$$a = x^t_{i,j} = y^t_{i',j'} \quad \text{and} \quad b = x^t_{i',j'} = y^t_{i,j}$$

be the two elements in Y_t that are at the "wrong" positions (relative to X_t), i.e., X_t and Y_t differ by a transposition of a and b.

If $i = i'$, i.e., a and b are in the same row, construct Y_{t+1} by moving b to a's position in X_{t+1}, a to b's position and put the remaining elements in their positions from X_{t+1}. Since the permutation of the row i was uniform in X_t, we likewise get that the permutation in



(a) M **(b)** M.Stir(1) **(c)** M.Stir(2) **(d)** M.Stir(3) **(e)** M.Stir(4) **(f)** M.Disperse()

Figure 1: An example of a single megaround of the WhiskRS shuffle for $k = 4$ and $n = 16$. Subfigure (a) shows a 4×4 matrix M containing 16 elements. Subfigures (b)–(e) show the results of the Stir(i) **operations for $i = 1, 2, 3, 4$, where each operation is executed independently and in parallel by a different server and randomly shuffles the ith row of the matrix. Subfigure (f) shows the public result of the** Disperse **operation.**

Y_t is uniform (and uniquely determined by the ith Stir in the chain X_t).

If $i \neq i'$, i.e., a and b are in different rows, then, they cannot switch in one step. The best we can hope for is that Disperse will map them both to the same row in X_{t+1} so that they can switch in the step after that. Notably, if we do not want to increase our distance, to each row in Y_t we must apply the same permutation (identical Stir) as we did in X_t, which was uniformly random in S_k. Therefore, with probability $1/k$, the elements a and b end up in the same row at time $t+1$. More precisely, the position of, say, a before Disperse in chain Y_t is fixed given the coins used for the transition $X_t \to X_{t+1}$ and since the permutation of a's row (the ith row) is uniform, there is probability $1/k$ that a lands in the position that gets moved by Disperse to the row b landed in in Y_{t+1} (recall from Section 2.2 that Disperse has, true it its name, the property that each element from row i at time t is mapped to a distinct row at time $t+1$). Once they have moved to the same row in Y_{t+1}, we can guarantee they switch at time $t+2$ and the two chains couple.

It follows that

$$\mathbf{E}[d(X_2, Y_2) \mid d(X_0, Y_0) = 1] = \begin{cases} 0 & \text{if } i = i', \\ 1 - \frac{1}{k} & \text{otherwise.} \end{cases}$$

We can therefore apply Lemma 2.6 with $\delta = 2$, $\eta = 1/k$, and $D = n - 1 = k^2 - 1$ to obtain, for a fixed ε:

$$\tau_{\text{mix}} = O\big(\log(D)/\eta\big) = O\big(k \log(k^2)\big) = O(k \log k) = O(\sqrt{n} \log n)\,.$$

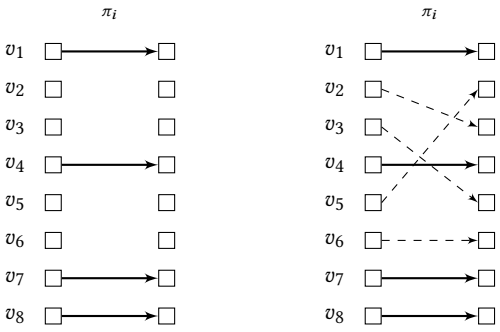
□

Note that the above analysis does not account for $\beta > 0$ and no security claim can be made in that case (cf. criticism of the square lattice shuffle [23] due to Larsen et al. [31]). In particular, if $\beta > k = \sqrt{n}$, the adversary can, in each megaround, target the particular row where a given ciphertext of interest is located and "trace" it throughout the entire permutation, winning the experiment in Definition 3.3 with probability 1. Also note that, while the different Stirs in WhiskRS *can* be done in parallel by the k shufflers (identified with block proposers; recall Section 2.2), they need not be: the last block proposer anyway has to wait until it is their slot to speak, so they may also delay the shuffling and adaptively bias their Stir. As we will see, for Whisk, where shuffling rounds are independent, β corruptions are naturally taken into account.

5 Analysis of Whisk

Recall from Section 2.2, that Whisk [27] relies on the distributed shuffling algorithm due to Larsen et al. [31]. In each of the t rounds, the shuffler for that round (Ethereum's block proposer for that slot) samples uniformly at random k out of n elements and permutes them (using a uniform permutation from S_k), proving in zero knowledge the correctness of this permutation and the corresponding rerandomization of the elements.

This section gives a Markov-chain analysis of Whisk's mixing time and argues that, as opposed to WhiskRS, it accounts for non-zero corruption budgets α and β (recall Definition 3.4).



(a) Public view **(b) Private view**

Figure 2: Two views of a single round of Whisk. Dashed links are hidden from the public and known only to the node performing the mixing (block proposer for the slot).

5.1 Mixing Time of Whisk

This section elaborates how Whisk can be viewed as a general case of Randomized Partial Checking (RPC) introduced by Jakobsson et al. [26] as a way of verifying correctness in mixnets. In RPC, t servers are privately permuting a set of n ciphertexts in t rounds, rerandomizing the ciphertexts each time. Every round t, correctness of the shuffle must be checked and so some subset of links of the permutation π_t, i.e., pairs $(i, \pi_t(i))$, gets opened with each link opened independently with some probability p.

Whisk can be conceived of as a variant of RPC where a fixed-size subset of links to be opened is sampled (equivalently: $n - k$ links are sampled without replacement instead of independently). Specifically, since only k trackers are shuffled each slot, we can extend the permutation $\tilde{\pi}_t : [k] \to [k]$ chosen (uniformly at random) by the tth shuffler to $\pi_t : [n] \to [n]$ by setting π_t to be the identity map on the remaining $n - k$ trackers. In other words,

the links on the idle $n-k$ trackers are "open" and visible to everyone (see Figure 2a).

Before proceeding with the details, observe that one may perform a simplified analysis of Whisk by relying on the results about mixing time for RPC [34] setting $1-p = k/n$, i.e., having the probability that a link is *not* revealed be proportional to the size of the shuffle batch k. Denote by $\mathsf{RPC}_{t,n,p}$ the permutation on n elements after t rounds of RPC with each link opened independently with probability $p \in (0,1)$. Then, we have the following result,

Lemma 5.1 (Lemma 6 of [34]). *If the number of rounds t satisfies*

$$t \geq t(n,p,\varepsilon) = \frac{\log\binom{n}{2} - \log\varepsilon}{\log\left(\frac{1}{1-(1-p)^2}\right)},$$

then the separation

$$s\big(\mathsf{RPC}_{t,n,p}, \mathcal{U}(S_n)\big) \leq \varepsilon\,.$$

Applying Lemma 5.1 to Whisk with $1-p = k/n$ gives satisfactory results for large values of n and $k = \sqrt{n}$. The reason for this will become clear once we perform the careful analysis, which we proceed to do next.

We show a more general result than Lemma 5.1 following the proof technique of [34] and [33]. We shall consider *pairs* of trackers and study the event that a pair i, j "moves unbeknownst to an adversary." Specifically, we shall consider the time T_{ij} that the trackers i, j both fall into the shuffle batch (of size k) for the first time. Since the permutation on each batch is uniformly random, the relative position of i, j at times $t > T_{ij}$ is uniformly random. Taking the maximum over all pairs i, j will give us a strong stationary time T of the underlying Markov chain that we will use to bound the separation of Whisk and a uniform distribution on S_n [2].

Before proceeding with the statement of the theorem, we comment on the malicious-adversary model (recall Section 3), where an adversary has statically corrupted up to α trackers and can adaptively corrupt up to β shufflers. Since each shuffling round in Whisk (and our generalized RPC) is independent of all others, it suffices to determine the number of "honest" rounds T needed to achieve the target separation ε and, in the end, set the total number of rounds to $T + \beta$ (cf. also results from [31]), since the adversary cannot "strategize" and choose shufflers to corrupt in any better way than at random. As for the α corrupted trackers, observe that, at the strong stationary time T, when all n trackers are permuted uniformly, so are in particular the $n - \alpha$ "honest" (uncorrupted) trackers. It therefore suffices to consider the case with no corruptions ($\alpha = 0$). Compare this again with Larsen et al. [31] who, in their model, fix a bound α on number the number of corruptions but then require the claim to hold for any number of corruptions less than α. Since the bound is expressed solely in terms of the bound α and not the actual number of corruptions, their claim sometimes holds vacuously.

Theorem 5.2. *Let $\mathsf{Whisk}_{t,n,k}$ be the permutation achieved by running Whisk (generalized RPC) on n trackers for t rounds, where each round randomly selects a subset of size k and permutes it uniformly at random. If the number of rounds t is*

$$t \geq t(n,k,\varepsilon) = \frac{\log\binom{n}{2} - \log\varepsilon}{\log\left((1-\frac{k}{n}\cdot\frac{k-1}{n-1})^{-1}\right)},$$

then the separation (as well as the total variation distance, recall Section 2.3)

$$s\big(\mathsf{Whisk}_{t,n,k}, \mathcal{U}(S_n)\big) \leq \varepsilon\,.$$

Proof. For clarity, identify the set of n trackers with $[n] = \{1,\ldots,n\}$. Write Z_t for the batch sampled for shuffling in round t, i.e., Z_t is uniformly distributed in subsets of $[n]$ of size k. Fix some trackers $i, j \in [n]$. We are interested in finding the time T_{ij} needed for this pair to be "mixed," i.e., the time their relative position becomes unknown to the adversary:

$$T_{ij} = \min\{t : i, j \in Z_t\}\,.$$

Note that for all $t > T_{ij}$, the relative position of i and j remains unknown to the adversary. Consider the maximum over all pairs i, j of T_{ij}:

$$T = \max_{i,j}\{T_{ij}\}\,.$$

If we view Whisk as a Markov chain

$$(\mathsf{Whisk}_{0,n,k}, \mathsf{Whisk}_{1,n,k}, \ldots)$$

with state space S_n, we see that T is a strong stationary time of this chain (compare with [33, 34]) and, by Lemma 2.4,

$$s\big(\mathsf{Whisk}_{t,n,k}, \mathcal{U}(S_n)\big) \leq \Pr[T > t]\,. \tag{1}$$

Consider the probability that the two trackers i, j "move" in round t unbeknownst to the adversary, i.e., are in Z_t (were selected for the shuffle at time t):

$$\Pr[i,j \in Z_t] = \frac{\binom{n-2}{k-2}}{\binom{n}{k}} = \frac{k}{n}\cdot\frac{k-1}{n-1}\,.$$

Then

$$\Pr[T_{ij} > t] = \prod_{\tau=1}^{t}(1 - \Pr[i,j \in Z_t]) = \left(1 - \frac{k}{n}\cdot\frac{k-1}{n-1}\right)^t.$$

Combining the above with Equation (1) and applying a union bound gives us that

$$\begin{aligned} s\big(\mathsf{Whisk}_{t,n,k}, \mathcal{U}(S_n)\big) &\leq \Pr[T > t] \\ &= \Pr\left[\bigcup_{i\neq j}\{T_{ij} > t\}\right] \\ &\leq \sum_{i\neq j}\Pr[T_{ij} > t] \\ &= \binom{n}{2}\left(1 - \frac{k}{n}\cdot\frac{k-1}{n-1}\right)^t. \end{aligned}$$

Setting the final expression to ε and solving for t yields the claim. □

It becomes evident why the approximation using RPC with $1 - p = k/n$ works. The only difference is the term $(k-1)/(n-1)$ in the denominator instead of another k/n (compare with Lemma 5.1).

Importantly, Theorem 5.2 gives an *upper bound* on the mixing time: if we run Whisk for at least $t(n,k,\varepsilon)$ rounds, we need not run it any further. The implication is, however, one-way: it could be that for a smaller number of rounds t, mixing is still good, but there is no justification we could find for this.

5.1.1 Implications for Ethereum. Note how, for $n = 2^{14}$ and $k = 128$ as proposed by the Ethereum community [27] and $\varepsilon = 2^{-40}$ as in [10, 11, 32], we get $t \geq 2^{19}$, while the proposal currently has $t = 2^{13} - 32$ instead. The confidence of the proposal authors seems to stem from a false claim in [31]. Indeed, Table 1 in [31] reports the number of honest shuffling rounds needed for different batch sizes k. The results, however, seem to be based on a theoretical result [31, Theorem 2] which importantly assumes $k \geq 256 \log_2(n/\delta)/(1 - \alpha/n)^2$, where δ is the probability of the shuffle failing (see discussion in Section 3). This assumption is violated when setting $k = 128$ and the claim is then completely unjustified. Our analysis in this work is therefore the first rigorous treatment of the anonymity guarantees of Whisk.

5.1.2 Alternative Approach. We briefly sketch how one can arrive at a slightly better bound via the same approach as in Section 4, namely, via path coupling. Consider adjacent states X_0 and Y_0 differing by a single transposition of two elements, call them a and b. At each time t, X_t evolves by choosing a subset $Z_t \subseteq [n]$ of cardinality k and permuting it with a uniformly random permutation σ_t of Z_t. If $\{a, b\} \not\subseteq Z_t$, apply σ_t to Y_t as well; otherwise, swap a and b, and then apply π_t. In the latter case, X_t and Y_t will have coupled. We can apply Lemma 2.6 with $\delta = 1$, $D = n - 1$, and

$$\eta = \Pr[a, b \in Z_t] = \frac{k}{n} \cdot \frac{k-1}{n-1}$$

to get, for any $\varepsilon \in (0, 1)$,

$$\tau_{\text{mix}}(\varepsilon) \leq \frac{n(n-1)}{k(k-1)} \left(\log(n-1) - \log \varepsilon\right) .$$

The bounds obtained by the two methods are compared in Figure 3.

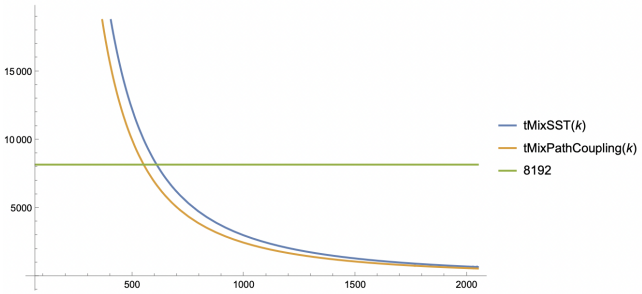


Figure 3: Comparison of mixing time bounds for Whisk due to the SST analysis and the path coupling analysis with batch size k on the x-axis and the mixing time for $n = 2^{14}$ and $\varepsilon = 2^{-40}$ on the y-axis. The green line set at 8192 corresponds to the number of rounds of Whisk in Ethereum.

5.2 Security of Whisk

Theorem 5.3. *Let $t = 2^{13} - 32$, $n = 2^{14}$ (as in the Whisk proposal [27]), and $\beta = 5209$. Then, for $k = 2048$, Whisk is a $(2^{-40}, \lambda)$-secure (t, n, k)-shuffle in the presence of any (α, β)-adversary (Definition 3.4).*

(Note that $\beta/t \approx 64\%$, i.e., we allow for more than a half of the shufflers to be corrupted.)

Proof. Let $\mathcal{A}$ be an (α, β)-adversary as in Definition 3.4 and write T for the strong stationary time (Definition 2.3) of the Markov chain $\text{Whisk}_{\tau,n,k}$. Note that shuffling rounds in Whisk are independent, i.e., for each pair of trackers i, j, the probability that i and j get swapped is the same for all times τ (and equal $(n-1)!/2n!$), regardless of whether one conditions on previous rounds.[10] Because of this, the β corruptions available to the adversary can just be accounted for by considering the number of "honest" rounds $t' = t - \beta$, as already done by Larsen et al. [31]. Indeed, since the adversary cannot strategically choose shufflers to corrupt, β's effect on the overall mixing time is additive.

Say $\mathcal{A}$ wins if, in the event conditioned on in Definition 3.4, it returns a pair (i, j) with $\pi(i) = j$ and write $m = n - |C \cup Q|$ for the number of "honest" (i.e., uncorrupted and not queried) indices after $\mathcal{A}$ has halted. Then

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins}] = \Pr[\mathcal{A} \text{ wins} \mid T > t'] \cdot \Pr[T > t'] \\ + \Pr[\mathcal{A} \text{ wins} \mid T \leq t'] \cdot \Pr[T \leq t'] ,\end{aligned}$$

or, equivalently,

$$\Pr[\mathcal{A} \text{ wins}] = \Pr[\mathcal{A} \text{ wins} \mid T \leq t] + \Pr[T > t] \cdot B ,$$

where $B = \Pr[\mathcal{A} \text{ wins} \mid T > t] - \Pr[\mathcal{A} \text{ wins} \mid T \leq t]$ and so $-1 \leq B \leq 1$. Also, by definition of a strong stationary time (Definition 2.3), the probability that $\mathcal{A}$ wins if we are past T is exactly $1/m$. Indeed, for times $\tau \geq T$, $\text{Whisk}_{\tau,n,k}$ is uniform on S_n and so is in particular uniform on the m remaining (uncorrupted and not queried) indices. On the other hand, by the proof of Theorem 5.2, $\Pr[T > t'] \leq \varepsilon(t')$, where

$$\varepsilon(t') := \binom{n}{2} \left(1 - \frac{k}{n} \cdot \frac{k-1}{n-1}\right)^{t'} .$$

It follows that

$$\left| \Pr[\mathcal{A} \text{ wins}] - \frac{1}{m} \right| \leq \varepsilon(t') .$$

(Accounting for the negligible leakage due to computational secrecy of encryption or computational hiding of commitments can be done the same way as done by Larsen et al. [31].)

The claim follows by substituting the numerical values $n = 2^{14}$, $k = 2048$, and $t' = t - \beta = 2951$. □

Due to the technical limitations of the zero-knowledge proof system used in Whisk (Curdleproofs), we cannot have k not a power of two. On the other hand, for $k = 1024$, we can only bound the TVD to be at most 2^{-19} (see Table 1), for $\beta = 0$ even, which is we believe to be insufficient.

6 Conclusions

We summarize, in Table 1, the maximum values of total variation distance away from uniformity of the Whisk permutation for the Ethereum parameters, namely, $n = 2^{14}$ elements and $t = 2^{13} - 32$ shuffling rounds, and for different values of the batch size k. Note that, when $n = k$, we obtain (by definition but also by the bound in proof of Theorem 5.2) a perfectly uniform distribution after a single shuffling round, i.e., $\text{Whisk}_{1,n,n} \equiv \mathcal{U}(S_n)$.

As mentioned previously, ours are the first rigorous analyses of the actual anonymity afforded by Whisk, and it can be seen

[10] In particular, $\text{Whisk}_{\tau,n,k}$ is a time-homogeneous Markov chain.

Table 1: Total variation distance between the uniform distribution $\mathcal{U}(S_n)$ and $\mathrm{Whisk}_{t,n,k}$ for $n = 2^{14}$ elements, $t = 2^{13} - 32$ rounds, and different values of $k \in \{128, 256, 512, 1024, 2048, n\}$.

k	128	256	512	1024	2048	$n = 2^{14}$
TVD	≤ 1	≤ 1	≤ 1	$\leq 2^{-19}$	$\leq 2^{-159}$	0

in Table 1 that the batch size $k = 128$ that the Ethereum community chose to settle on is likely too small (for n and t fixed as in the Whisk proposal [27]). More precisely, as argued in Sections 3 and 5, there is nothing in the way of theoretical justification for this choice of parameters. Importantly, however, our work does not establish that the choice $k = 128$ admits a practical attack. We nevertheless strongly urge the Ethereum community to revise the Whisk proposal [27] and clearly state the trade-off between block proposal efficiency and a conservative, theoretically justified, batch size $k \geq 2048$.

The Ethereum community has settled on $k = 128$ primarily due to the overhead of the underlying zero-knowledge argument system (recall Sections 3 and 5.1), with the current proposal relying on Curdleproofs [29]. We have run the benchmarks from its official GitHub repository [14] for different values of k on an Intel(R) Core(TM) Ultra 7 165H @ 3.80 GHz, a mid-range laptop CPU consistent with current hardware recommendations. Indeed, while Ethereum's official documentation [18] lists the minimum requirements for a node to be a "fast CPU with 4+ cores," more user-focused guides, for instance [13], suggest using a "processor with at least 8 cores / 16 threads, ideally around 3.5 GHz per core, to handle block validation during peak loads." We observe approximately a twofold improvement in performance over what is reported in the Curdleproofs repository [14]. Moreover, for the proposed value of $k \approx 2048$, the observed running time remains practical at approximately 2 seconds for the prover. See Table 2 for detailed measurements.

Table 2: Mean execution times (ms) over 100 runs of the Curdleproofs benchmark on an Intel(R) Core(TM) Ultra 7 165H @ 3.80 GHz for $n = 2^{14}$ and $k+4 \in \{128, 256, 512, 1024, 2048, n\}$ (four trackers are reserved for internal use by the Curdleproofs system.

k	124	252	508	1020	2044	$n = 2^{14} - 4$
Shuffling	38	86	162	324	595	7 468
Proving	165	305	548	1 046	2 034	23 647
Verifying	15	18	23	30	46	381

6.1 Future Work

To conclude, we outline a number of directions for future work. First, lower bounds on the mixing time for Whisk or WhiskRS are missing to complement Theorems 4.1 and 5.2 and formally establish that the parameter choices currently proposed for Ethereum lead to insecure shuffling. Exactly formalizing the necessary conditions under which β shuffler corruptions can be accounted for by extending the mixing time by β additively would also be needed to salvage WhiskRS. We note that the conditions identified in the proof of Theorem 5.2, namely, independence of any shuffle both from the time τ and previous shuffles, are *sufficient*, but need not be *necessary* and weaker conditions may be enough. Establishing these weaker conditions would facilitate proposing and reasoning about novel distributed shuffling protocols that, e.g., leverage parallelism (as does WhiskRS) or make explicit use of the blockchain environment in which they are run. Finally, evaluating state-of-the-art shuffling arguments other than Curdleproofs could show larger values of k in Whisk are feasible.

Generative AI Usage. Some text was edited for grammar using Grammarly. All AI-assisted content has been carefully checked and verified.

References

[1] David Aldous and Persi Diaconis. 1986. Shuffling cards and stopping times. *The American Mathematical Monthly* 93, 5 (1986), 333–348.

[2] David Aldous and Persi Diaconis. 1987. Strong uniform times and finite random walks. *Advances in Applied Mathematics* 8, 1 (1987), 69–97.

[3] Stephanie Bayer and Jens Groth. 2012. Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In *Advances in Cryptology – EUROCRYPT 2012*, David Pointcheval and Thomas Johansson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 263–280.

[4] Mihir Bellare, Joe Kilian, and Phillip Rogaway. 1995. The Security of the Cipher Block Chaining Message Authentication Code. In *Advances in Cryptology — CRYPTO '94 (LNCS, Vol. 839)*. Springer, 341–358. doi:10.1007/3-540-48658-5_32

[5] Dan Boneh, Saba Eskandarian, Lucjan Hanzlik, and Nicola Greco. 2020. Single Secret Leader Election. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT)*. 12–24.

[6] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. 2014. Mixcoin: Anonymity for Bitcoin with Accountable Mixes. In *Financial Cryptography and Data Security (LNCS, Vol. 8437)*. Springer, 486–504.

[7] Russ Bubley and Martin Dyer. 1997. Path coupling: A technique for proving rapid mixing in Markov chains. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*. IEEE, 223–231.

[8] Tereza Burianová, Martin Perešíni, and Ivan Homoliak. 2025. Secret Leader Election in Ethereum PoS: An Empirical Security Analysis of Whisk and Homomorphic Sortition under DoS on the Leader and Censorship Attacks. arXiv:2509.24955 [cs.CR] https://arxiv.org/abs/2509.24955

[9] Vitalik Buterin. 2014. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. Accessed: 12 January 2026. https://ethereum.org/en/whitepaper

[10] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. 2019. Two-Party ECDSA from Hash Proof Systems and Efficient Instantiations. In *Advances in Cryptology – CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III* (Santa Barbara, CA, USA). Springer-Verlag, Berlin, Heidelberg, 191–221. doi:10.1007/978-3-030-26954-8_7

[11] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. 2020. Bandwidth-Efficient Threshold EC-DSA. In *Public-Key Cryptography – PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part II* (Edinburgh, United Kingdom). Springer-Verlag, Berlin, Heidelberg, 266–296. doi:10.1007/978-3-030-45388-6_10

[12] David L Chaum. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24, 2 (1981), 84–90.

[13] Cherry Servers. 2025. *Ethereum Node Hardware Requirements (2025 Edition)*. Accessed: 2026-01-15. https://www.cherryservers.com/blog/ethereum-node-requirements?utm_source=chatgpt.com

[14] Curdleproofs Contributors. 2023. Curdleproofs. https://github.com/asn-d6/curdleproofs.

[15] Jean-Sébastien Coron, Yevgeniy Dodis, and C. Shrimpton. 2003. Security Analysis of Feistel Networks. In *Advances in Cryptology — CRYPTO 2003 (LNCS, Vol. 2729)*. Springer, 254–271. doi:10.1007/978-3-540-45146-4_16

[16] Artur Czumaj, Przemka Kanarek, Mirosław Kutyłowski, and Krzysztof Loryś. 1999. Delayed path coupling and generating random permutations via distributed stochastic processes. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Baltimore, Maryland, USA) *(SODA '99)*. Society for Industrial and Applied Mathematics, USA, 271–280.

[17] Persi Diaconis and Mehrdad Shahshahani. 1981. Generating a Random Permutation with Random Transpositions. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete* 57, 2 (1981), 159–179.

[18] Ethereum Foundation. 2025. *Spin up your own Ethereum node.* Accessed: 2026-01-15. https://ethereum.org/developers/docs/nodes-and-clients/run-a-node/

[19] Ben Fisch, Arthur Lazzaretti, Zeyu Liu, and Charalampos Papamanthou. 2024. ThorPIR: Single Server PIR via Homomorphic Thorp Shuffles. Cryptology ePrint Archive, Paper 2024/482. https://eprint.iacr.org/2024/482

[20] Scott Fluhrer, Itsik Mantin, and Adi Shamir. 2001. Weaknesses in the key scheduling algorithm of RC4. In *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16–17, 2001 Revised Papers 8*. Springer, 1–24.

[21] Yue Gao, Jinqiao Shi, Xuebin Wang, Ruisheng Shi, Zelin Yin, and Yanyan Yang. 2021. Practical Deanonymization Attack in Ethereum Based on P2P Network Analysis. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. 1402–1409. doi:10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00191

[22] Jens Groth and Yuval Ishai. 2008. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 379–396.

[23] Johan Håstad. 2006. The square lattice shuffle. *Random Structures and Algorithms* 29, 4 (2006), 466–474.

[24] Lioba Heimbach, Yann Vonlanthen, Juan Villacis, Lucianna Kiffer, and Roger Wattenhofer. 2024. Deanonymizing Ethereum validators: The P2P network has a privacy issue. *arXiv preprint arXiv:2409.04366* (2024).

[25] Viet Tung Hoang and Stefano Tessaro. 2016. The Multi-user Security of Symmetric Encryption. In *Advances in Cryptology — CRYPTO 2016 (LNCS, Vol. 9815)*. Springer, 381–408. doi:10.1007/978-3-662-53008-5_14

[26] Markus Jakobsson, Ari Juels, and Ronald L Rivest. 2002. Making mix nets robust for electronic voting by randomized partial checking.. In *USENIX security symposium*. San Francisco, USA, 339–353.

[27] George Kadianakis. 2022. Whisk: A practical shuffle-based SSLE protocol for Ethereum. https://ethresear.ch/t/whisk-a-practical-shuffle-based-ssle-protocol-for-ethereum/11763.

[28] George Kadianakis and Justin Drake. 2024. EIP-7441: Upgrade block proposer election to Whisk. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-7441.md/.

[29] Georgios Kadianakis, Justin Drake, David Feist, Greg Herold, Dmitry Khovratovich, Mary Maller, Mike Simkin, and the Ethereum Foundation Cryptography Research Team. 2022. *Curdleproofs: A Shuffle Argument Protocol.* Technical Report. Ethereum Foundation Cryptography Research. Accessed: January 13, 2026.

[30] Dmitry Khovratovich. 2022. Privacy Analysis of Whisk. https://github.com/khovratovich/whisk-analysis/blob/main/WhiskAnalysis.pdf.

[31] Kasper Green Larsen, Maciej Obremski, and Mark Simkin. 2023. Distributed shuffling in adversarial environments. In *4th Conference on Information-Theoretic Cryptography (ITC 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[32] Yehuda Lindell. 2017. Fast Secure Two-Party ECDSA Signing. In *Advances in Cryptology – CRYPTO 2017*, Jonathan Katz and Hovav Shacham (Eds.). Springer International Publishing, Cham, 613–644.

[33] Paweł Lorek, Michał Kulis, and Filip Zagórski. 2017. Leakage-Resilient Riffle Shuffle. In *International Conference on Mathematical Aspects of Computer and Information Sciences*. Springer, 395–408.

[34] Paweł Lorek, Moti Yung, and Filip Zagórski. 2024. Mirrored Commitment: Fixing "Randomized Partial Checking" and Applications. In *International Conference on Applied Cryptography and Network Security*. Springer, 3–27.

[35] Ilya Mironov. 2002. (Not So) Random Shuffles of RC4. In *Advances in Cryptology – CRYPTO 2002 (LNCS, Vol. 2442)*. Springer, 304–319. doi:10.1007/3-540-45708-9_20

[36] Ben Morris and Phillip Rogaway. 2014. Sometimes-Recurse Shuffle. In *Advances in Cryptology – EUROCRYPT 2014 (LNCS, Vol. 8441)*. Springer, 311–326.

[37] C. Andrew Neff. 2001. A Verifiable Secret Shuffle and Its Application to E-Voting. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*. 116–125.

[38] Andreas Pfitzmann and Marit Köhntopp. 2001. *Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–9. doi:10.1007/3-540-44702-4_1

[39] Kazue Sako and Joe Kilian. 1995. Receipt-Free Mix-Type Voting Scheme. In *Advances in Cryptology – EUROCRYPT '95 (LNCS, Vol. 921)*. Springer, 393–403.

[40] Mathy Vanhoef and Frank Piessens. 2015. All Your Biases Belong to Us: Breaking {RC4} in {WPA-TKIP} and {TLS}. In *24th USENIX Security Symposium (USENIX Security 15)*. 97–112.

A Open Science

There are no artifacts needed to evaluate the paper's core contributions.

B General Security Model with Parametrized Number of Post-Shuffle Queries

Say $\mathcal{A}$ is an (α, β, γ)-*adversary* if $\mathcal{A}$ may corrupt (track) up to α ciphertexts, corrupt up to β shufflers (rounds), and query the resulting permutation π up to γ times (for forward or backward mappings).

Definition B.1 (Security [Generalized]). A (t, n, k)-shuffle Π is (ε, λ)-secure in the presence of a probabilistic polynomial-time (α, β, γ)-adversary $\mathcal{A}$, if for all $C \subset [n] : |C| \leq \alpha \leq n - 2$:

$$\left| \frac{1}{n - |C \cup Q|} - p \right| \leq \varepsilon + \mathsf{negl}(\lambda)\,,$$

where

$$p = \Pr\left[\pi(i) = j \;\middle|\; \begin{array}{l} (\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)\ \forall i \in [n] \\ c_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}_i, i)\ \forall i \in [n] \\ \mathbf{c} \leftarrow (c_1, \ldots, c_n) \\ \mathbf{sk} \leftarrow \{\mathsf{sk}_i \mid i \in C\} \\ (\pi, z) \leftarrow\!\!\$\ \langle \Pi(\mathbf{c}), \mathcal{A}(\mathbf{c}, \mathbf{sk}) \rangle \\ Q \leftarrow \emptyset \\ (i, j) \leftarrow\!\!\$\ \mathcal{A}^{O_\pi, O_{\pi^{-1}}}(z) \\ i \notin C \cup Q \end{array}\right]$$

and the probability is taken over the random coins of all algorithms involved. The oracles $O_\pi, O_{\pi^{-1}}$ are defined as follows.

$O_\pi(i)$:

1: **if** $|Q| = \gamma$ **then**
2: **return** $\perp$
3: $j \leftarrow \pi(i)$
4: $Q \leftarrow Q \cup \{i\}$
5: **return** j

$O_{\pi^{-1}}(j)$:

1: **if** $|Q| = \gamma$ **then**
2: **return** $\perp$
3: $i \leftarrow \pi^{-1}(j)$
4: $Q \leftarrow Q \cup \{i\}$
5: **return** i