# Distributed Shuffling in Adversarial Environments

Kasper Green Larsen[1⋆], Maciej Obremski[2⋆⋆], and Mark Simkin[3⋆⋆⋆]

[1] Aarhus University
[2] National University of Singapore
[3] Ethereum Foundation

**Abstract.** We formalize and study the problem of distributed shuffling in adversarial environments. In this setting, there are $m$ shufflers that have access to a public bulletin board that stores a vector $(c_1, \ldots, c_n)$ of re-randomizable commitments. The shufflers repeatedly read $k$ of the $n$ commitments, with $k$ potentially much smaller than $n$, and shuffle them. An adversary has the ability to initially corrupt and then track some of the commitments throughout the shuffles and can adaptively corrupt a bounded number of shufflers in every single round. The goal of the distributed shuffling protocol is to hide the output locations of commitments that are not corrupted by the adversary.

We present and analyze a protocol that solves this problem with essentially optimal shuffling complexity. As an exemplary data point, our protocol can shuffle a list of length $n$ with shuffles of size $k$, where $k \in \Omega(\lg^2 n)$, in the presence of an adversary that can corrupt $4n/5$ many shufflers in each round and can corrupt $4n/5$ commitments in the input vector. Our $m$-party shuffling protocol with $m \in \Omega(n/k)$ terminates in $\mathcal{O}(\lg n)$ rounds. We provide numerical benchmarks that validate our theoretically proven guarantees and in fact show that the number of rounds is not just theoretically, but also concretely small.

Our shuffling protocol can either improve efficiency or lead to more secure solutions in multiple research domains, such as the design of mix-nets, single secret leader election protocols, and electronic voting.

## 1 Introduction

Shuffling the elements of a long vector efficiently is a problem that appears in various shapes and forms throughout many different domains of cryptography. In anonymous communication systems [Cha81, SK95, JJR02] a set of senders would each like to communicate one messages to a set of receivers without revealing who is talking to who. In electronic voting [SK95, JJR02, Nef01], we have a long list of votes and we would like to determine the election outcome without revealing who voted for who. In the domain of cryptocurrencies [Max13, BNM+14], we have multiple payers, who would like to transfer money to multiple payees without revealing who is paying who. One popular approach, which dates back to Chaum's original work [Cha81], for achieving anonymity in all of the above applications, is through the use of so called mix-nets. The main idea of mix-nets is to let one or more semi-trusted shufflers read the whole input vector, shuffle it locally, and send it to the next shuffler before eventually publishing the output vector that has been shuffled by multiple entities. From a security point of view, this approach provides strong anonymity guarantees. As long as only a single shuffler is honest, the order of elements in the output vector looks completely random to any adversarial observer.

From an efficiency perspective, however, such strong anonymity guarantees do not come for free. The computational and the bandwidth overhead of each shuffler grows linearly in the length of the

⋆ larsen@cs.au.dk. Supported by Independent Research Fund Denmark (DFF) Sapere Aude Research Leader grant No 9064-00068B and a Villum Young Investigator grant.
⋆⋆ obremski.math@gmail.com. Funded by MOE2019-T2-1-145 Foundations of quantum-safe cryptography.
⋆⋆⋆ mark.simkin@ethereum.org

vector that should be shuffled. In applications like electronic voting, the length of a vector of votes could easily be in the millions, which places a significant burden on each shuffler. In addition, many applications also require each shuffler to provide a zero-knowledge proof attesting the correctness of the performed shuffle [SK95, FS01, Nef01, BG12], which incurs a large computational cost.

Mix-nets are not the only approach for achieving anonymity in applications, such as the ones mentioned above. Alternatives, like onion routing schemes [GRS96, RSG98, DMS04], do exist and can provide much better performance, but often do so at the cost of weaker anonymity guarantees [MD05, MZ07, BMG+07]. If a maximum degree of anonymity is required, then mix-nets are a very attractive approach.

### 1.1 Our Contribution

In this work, we formalize and study the problem of distributed shuffling in adversarial environments. Here, we have $m$ shufflers and a public bulletin board, visible to everyone, that stores a vector $(c_1, \ldots, c_n)$ of re-randomizable commitments[4]. A distributed shuffling protocol is an interactive process by which all shufflers repeatedly read $k$ of the $n$ commitments, locally re-randomize and shuffle them, and then upload them back to the bulletin board. At the end of the protocol execution, a permutation of the original vector of commitments should be written on the bulletin board. At the start of a protocol execution, adversary $\mathcal{A}$ is allowed to corrupt a subset of indices $I \subset \{1, \ldots, n\}$ with $|I| \leq \alpha$ and can then track all commitments $c_i$ for $i \in I$ throughout the shuffling process. $\mathcal{A}$ can actively corrupt *different subsets consisting of $\beta$ shufflers adaptively in every single* round of the protocol. Finally, $\mathcal{A}$ even gets to see $\gamma$ additional values in the output vector and where they came from at the end of the protocol execution.

We present a distributed shuffling protocol, which guarantees that the adversary's probability of guessing the location of a commitment $c_i$ with $i \notin I$ that was not among the $\gamma$ opened commitments, is only a small constant factor better than random guessing. The main result of our work is a general theorem that captures the relationship between the number of parties, the size of each local shuffle, the required number of rounds in our shuffling protocol and the various corruptions that the adversary can perform. The following informal theorem is a corollary of our main result for a parameter setting that allows the adversary to corrupt large constant fractions of commitments in the input vector, shufflers per round, and commitments in the output vector.

**Theorem 1 (Informal).** *Let $\beta = 4n/5$ be the number of corruptions per round, let $\alpha \leq 4n/5$ be the number of corrupted commitments in the input vector, and let $\gamma \leq 4n/5$ be the number of commitments in the output vector, whose location in the input vector is revealed. For a universal constant $C$, for any $k \geq C \cdot (\lg^2 n)$, there exists a secure distributed shuffling protocol among $m \in \Omega(n/k)$ parties that runs in $\mathcal{O}(\lg n)$ rounds.*

The theorem above shows that even in highly adversarial settings our protocol can quickly, i.e. on the order of $\lg n$ rounds, shuffle the elements of the input vector. It also shows that for certain parameter ranges, our protocol allows the the total workload of each individual shuffler to be *sublinear* in the length of the vector. To underline the practicality of our presented solution, we provide benchmarks, which show that the efficiency of our protocol is not only asymptotic, but also concrete.

[4] Throughout the paper we focus on commitments, but all of our results apply equally well to ciphertexts of a (re-randomizable) encryption scheme.

Lastly, we discuss several applications, such as mix-nets, single secret leader elections, and electronic voting, that can benefit from our new shuffling protocol in terms of either improved security or improved efficiency.

### 1.2 Other Related Works

A series of existing works [Tho73, BD92, Hås06, Hås16, RY13, MR14] has studied the question of how long it takes to shuffle the elements of a vector via either smaller or restricted shuffling operations. There, the problem is studied in a benign setting without an adversarial presence. All of these shuffling algorithms immediately fail in the presence of an adaptive adversary and can thus not be translated to solve the problem considered in this work.

## 2 Preliminaries

**Notation.** We write $[n]$ to denote the set $\{1, \ldots, n\}$. For a set $X$, we write $x \leftarrow X$ to denote the process of sampling a uniformly random element $x$ from $X$. For a randomized algorithm $A$ we write $A(x; r)$ to explicitly specify the random tape $r$ when $A$ is executed on some input $x$. Otherwise, we write $A(x)$ and simply assume that $r$ is implicitly chosen uniformly at random. We write $\perp \leftarrow A(x)$ to denote that an algorithm $A$ failed to produce an output. We define that for any $n \in \mathbb{N}$ and any function $f$ that $\perp + n := \perp$ and $f = \perp \implies f(n) = \perp$. We denote the computational security parameter by $\lambda$.

### 2.1 Commitment schemes

Here we recall the standard definition of a commitment scheme.

**Definition 1.** *A commitment scheme with message space $\mathcal{M}$ is a tuple of PPT algorithms* $(\mathtt{Setup}, \mathtt{Commit}, \mathtt{Open})$ *that are defined as follows:*

$\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$**:** *The setup algorithm takes the security parameter $1^\lambda$ as input and returns public parameters* $\mathsf{pp}$.

$(e, d) \leftarrow \mathtt{Commit}(\mathsf{pp}, m)$**:** *The commitment algorithm takes public parameters* $\mathsf{pp}$ *and a message $m \in \mathcal{M}$ as input and returns a commitment $e$ along with opening information $d$.*

$m \leftarrow \mathtt{Open}(\mathsf{pp}, e, d)$**:** *The opening algorithm takes the public parameters* $\mathsf{pp}$*, commitment $e$, and opening $d$ as input and returns $m \in \mathcal{M} \cup \{\perp\}$.*

**Definition 2 (Hiding).** *A commitment scheme* $(\mathtt{Setup}, \mathtt{Commit}, \mathtt{Open})$ *is computationally hiding, if for any adversary PPT $\mathcal{A}$ and for any two messages $v_1, v_2$, it holds that*

$$|\Pr[\mathcal{A}(\mathsf{pp}, \mathtt{Commit}(\mathsf{pp}, v_1) = 1] = \Pr[\mathcal{A}(\mathsf{pp}, \mathtt{Commit}(\mathsf{pp}, v_2) = 1]| \leq \mathsf{negl}(\lambda)\,,$$

*where the randomness is taken over the random coins of the setup algorithm, the adversary, and the commitment scheme.*

**Definition 3 (Binding).** *A commitment scheme* $(\mathtt{Setup}, \mathtt{Commit}, \mathtt{Open})$ *is perfectly biding, if for any PPT adversary $\mathcal{A}$, it holds that*

$$\Pr\left[(e, d_1, d_2, v_1, v_2) \leftarrow \mathcal{A}(\mathsf{pp}) : \begin{array}{r} \mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda) \\ \wedge\ \mathtt{Open}(\mathsf{pp}, e, d_1) = v_1 \\ \wedge\ \mathtt{Open}(\mathsf{pp}, e, d_2) = v_2 \\ \wedge\ v_1 \neq v_2 \end{array}\right] \leq \mathsf{negl}(\lambda)\,,$$

*Remark 1.* Throughout the paper we will omit the public parameters pp from the explicit inputs to the algorithms and assume that pp was generated in a trusted manner and is available to all shufflers.

For each shuffler to be able to read, re-randomize, and permute $k$ commitments locally, we need to assume that the commitments in the input vector for our distributed shuffling protocol satisfy the re-randomizability property defined below. We will never make explicit use of this property in our protocol descriptions, since we state our protocol in a fashion that is oblivious to the exact local shuffles that are being performed, but we provide the property here for completeness.

**Definition 4 (Perfect Re-Randomizability).** *We say a commitment scheme* (Setup, Commit, Open) *is perfectly re-randomizable, if there exists a PPT algorithm* Rand*, such that for any* $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda)$*, any message* $v$ *it holds that* $e$ *for* $(e, d) \leftarrow \mathtt{Commit}(\mathtt{pp}, v; r)$ *for a uniformly random* $r$ *is identically distributed to* $e' \leftarrow \mathtt{Rand}(e, r')$ *for a uniformly random* $r'$*.*

## 3 Model

In our model, we have a public bulletin board, where parties can post messages that are visible to all other parties. In the beginning, the only thing written on the message board are re-randomizable commitments $c_1, \ldots, c_n$ to distinct messages $v_1, \ldots, v_n$.

An $(m, n, k)$-shuffle $\Pi$ is a protocol among $m$ parties known as the shufflers. At the end of the protocol, the bulletin board will contain commitments $\tilde{c}_1, \ldots, \tilde{c}_n$ to some permutation of $v_1, \ldots, v_n$. The execution of $\Pi$ proceeds in rounds over a synchronized network. During each round, each shuffler reads $k$ commitments from the bulletin board and publish re-randomized and permuted versions thereof back on the board. Which commitments were shuffled by a shuffler is publicly visible. We stress two things. Firstly, during the protocol execution, multiple shufflers can read the same commitment from the board and publish independent copies of that commitment back on the board. Secondly, we assume that the commitments are (computationally) hiding in the sense that no PPT party other than the shuffler can determine the permutation between the read and the published commitments.

At the beginning of each round $i$, a randomness beacon $\mathcal{B}$ publishes an unpredictable uniformly random $\lambda$-bit string on the board. The shufflers can use it as an auxiliary input in rounds $j \geq i$ and can make their decisions of which $k$ commitments to shuffle dependent on these strings.

*Corruptions.* The shuffling protocol runs in the presence of adversarial behavior. The PPT adversary can see who posts which messages on the bulletin board and in addition can perform two types of corruptions. At the beginning of a protocol execution, before any shuffling is being performed, the adversary can corrupt $\alpha$ commitments. For each corrupted commitment (or re-randomized copy thereof), the adversary can see the corresponding values inside the commitment at any point in time during the protocol execution, thus it is effectively able to track these commitments. Additionally, the adversary is given a budget $\beta$ of adaptive corruptions of shufflers. In every round, after the value of the randomness beacon is revealed, the adversary can corrupt at most $\beta$ shufflers of its own choice for the duration of the round. We stress that a shuffler corrupted in round $t$ is not necessarily corrupted in round $t + 1$ and that the adversary may choose different sets of $\beta$ shufflers

that are corrupted in rounds $t$ and $t+1$ adaptively.[5] Shufflers that are corrupted by the adversary can perform arbitrarily malicious permutations on arbitrary choices of at most $k$ commitments.

We restrict maliciously corrupted shufflers to always be honest about which commitments they shuffled and to always honestly return a vector of valid commitments that commits to the same set as the vector of commitments that was read by that shuffler. In principle, we do not need to require either of those restrictions on the adversary's behavior, since both these guarantees can be enforced via standard zero-knowledge shuffling arguments. If the list were to contain pedersen commitments [Ped91], then efficient shuffling arguments of Bünz et al. [BBB+18] or Hoffmann et al. [HKR19] could be used. If the list were to contain ElGamal encryptions, then efficient shuffling arguments of Bayer and Groth [BG12] or Bünz et al. [BBB+18] could be used. For the sake of clarity, we chose to simply restrict the adversary to avoid talking about zero-knowledge arguments in this work.

*Efficiency metrics.* We measure the efficiency of a shuffling protocol through two metrics. The first one is the total number of shuffles performed during a protocol execution. The second one is the round complexity of the protocol.

*Definitions.* For a shuffling protocol $\Pi$ and an adversary $\mathcal{A}$, we write $(z, \pi) \leftarrow \langle \Pi(r), \mathcal{A}(\tilde{r}) \rangle$ to denote the execution $\Pi$ with random coins $r$ in the presence of $\mathcal{A}$ with random coins $\tilde{r}$, where $z$ is the adversary's output and $\pi$ is the permutation on domain $[n]$, i.e. between the input and output commitments' values. If at the end of a protocol execution the values inside the output commitments are not a permutation of the input commitents' values, then we write $\pi = \perp$.

**Definition 5 (Correctness).** *We say that an $(m, n, k)$-shuffle $\Pi$ is correct in the presence of an adversary $\mathcal{A}$, if*

$$\Pr[(z, \pi) \leftarrow \langle \Pi(r), \mathcal{A}(\tilde{r}) \rangle : \pi \neq \perp] = 1,$$

*where the probability is taken over the random coins $r$ and $\tilde{r}$.*

**Definition 6 (Security).** *Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be a stateful adversary that corrupts at most $\alpha$ commitments and $\beta$ shufflers. We say that an $(m, n, k)$-shuffle $\Pi$ is $(\alpha, \beta, \gamma, \delta, \epsilon)$-secure in the presence of a PPT adversary $\mathcal{A}$, if with probability at least $1 - \delta$ it holds that*

$$\Pr\left[\begin{array}{c}(z, \pi) \leftarrow \langle \Pi(r), \mathcal{A}_0(\tilde{r}) \rangle \\ (i, j) \leftarrow \mathcal{A}_1^{\mathcal{O}(\pi, \cdot)}(z)\end{array} : \pi(i) = j \wedge i \notin Q_\alpha \cup Q_\gamma\right] \leq \epsilon,$$

*where the randomness is taken over the random coins $r$ and $\tilde{r}$. The set $Q_\alpha$ with $|Q_\alpha| \leq \alpha$ is the set of indices of commitments that were initially corrupted and $Q_\gamma$ with $|Q_\gamma| \leq \gamma$ is the set of inputs that $\mathcal{A}_1$ submits to oracle $\mathcal{O}(\pi, \cdot)$, which is initialized with a counter $c = 1$ and, upon being triggered by $\mathcal{A}_1$, returns $\pi^{-1}(c)$ and sets $c = c + 1$ as long as $c \leq \gamma$ and otherwise returns $\perp$.*

## 4 Construction

In this section, we present our distributed shuffling protocol. Since our protocol relies on randomness beacons, we first formally define these in Section 4.1, before formally presenting our protocol in Section 4.2.

[5] This also means that, in principle, the shufflers in different rounds do not even need to be the same entities, but for the sake of simplicity we focus on shuffling as an $m$-party protocol.

### 4.1 One Randomness Beacon to Rule Them All

In the following, we will show that one randomness beacon that returns a $\lambda$-bit string in every round is sufficient to simulate multiple structured random outputs per round. Let us first define what we mean by a randomness beacon.

**Definition 7.** *Let $\mathcal{B} : \mathbb{N} \to \{0,1\}^\lambda$ be a randomness beacon that takes a time step $t \in \mathbb{N}$ as input and returns a $\lambda$-bit string. Let $\mathcal{A}$ be a PPT adversary, then for all $t \in \mathbb{N}$ it holds that*

$$\Pr\left[z \leftarrow \mathcal{A}(r^1, \ldots, r^t) : \begin{array}{r} r \leftarrow \mathcal{B}(i)\ \forall i \in [t] \\ \wedge\ \mathcal{B}(t+1) = z \end{array}\right] \leq \mathsf{negl}(\lambda)\,,$$

*where the probability is taken over the random choice of $\mathcal{B}$ and the random coins of $\mathcal{A}$.*

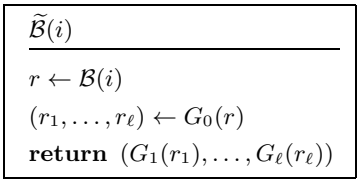


**Fig. 1.** Beacon with many outputs from beacon with one short output.

In the following theorem, we show that one can stretch a short pseudorandom seed returned by the randomness beacon into a long sequence of random outputs from different structured domains, if one is given PRGs with the appropriate output domains.

**Theorem 2.** *Let $\mathcal{B} : \mathbb{N} \to \{0,1\}^\lambda$ be a randomness beacon, let $G_i : \{0,1\}^\lambda \to Y_i$ for $i \in [\ell]$ and $G_0 : \{0,1\}^\lambda \to (\{0,1\}^\lambda)^\ell$ be secure pseudorandom generators. Then the construction in Figure 1 is a randomness beacon with output domain $Y_1 \times \cdots \times Y_\ell$.*

*Proof.* Let us start with $\mathsf{hybrid}_0$, which is defined like the game that the PPT adversary $\mathcal{A}$ is playing against randomness beacon $\widetilde{\mathcal{B}}$. Define $\mathsf{hybrid}_1$ identically to $\mathsf{hybrid}_0$ with the difference that all outputs of $G_0$ are replaced by uniformly random values. Indistinguishability of the two hybrids directly follows form the pseudo-randomness of $G_0$. We note that now in $\mathsf{hybrid}_1$, in every round, each PRG is called on an independent uniformly random input. Thus, pseudo-randomness of the output of $\widetilde{\mathcal{B}}$ directly follows form the security of all individual PRGs.

### 4.2 Distributed Shuffling Protocol

We will now present our main construction. We assume that the parties have access to multiple randomness beacons, which can all be simulated by one beacon as explained above. At the start of each round $t$ the beacon $\mathcal{B}_\pi$ outputs a uniformly random permutation $\pi^t$ over $[n]$, the beacon $\mathcal{B}_r$ outputs a uniformly random $\left(w_1^1, \ldots, w_1^T, w_2^1, \ldots, w_m^T\right) \in \mathbb{Z}_{n/k}^{T \cdot m}$, and the beacon $\mathcal{B}_s$ takes a vector $\left(\ell_1, \ldots, \ell_{n/k}\right)$ as input and returns a uniformly random vector $\left(b_1, \ldots, b_{n/k}\right) \in [\ell_1] \times \cdots \times \left[\ell_{n/k}\right]$. We note that in practice the beacon does not actually take any input, but instead the inputs, other than the time step, would simply define the domains of the PRGs that are applied to the beacon's

output. Furthermore, we will assume that all randomness beacons return perfectly random values as opposed to pseudo-random values to simplify the exposition of the proof of Theorem 5. It will be easy to see that all proofs and results carry over to randomness beacons with pseudo-random outputs.

In the description of our construction, we will make use of a method `Shuffle` that takes a vector of commitments $(\tilde{c}_1, \ldots, \tilde{c}_k)$ as input and returns a vector $(\hat{c}_1, \ldots, \hat{c}_k)$ of valid output commitments. The vector of messages inside the output commitments corresponds to a permutation of the messages inside the input vector and we assume that given the input and output vectors of commitments the permutation between them is computationally hidden.

**Protocol Intuition.** The main challenge that our protocol needs to overcome is the following. Assume a hypothetical distributed shuffling protocol that runs in $T$ rounds. In every round the shufflers take disjoint portions of size $k$ of the $n$ commitments and shuffle them separately. The result of this "light" shuffling constitutes the list of $n$ commitments that will be further shuffled in the proceeding rounds. Since the protocol runs for $T$ rounds and since no two shufflers read intersecting sets of commitments, it follows that none of the commitments in the initial list will be shuffled more than $T$ times in total. Furthermore, let us assume that the protocol, at the start, reveals who will shuffle what subset of $k$ commitments in each round. Now assume that an adversary $\mathcal{A}$ wants to ensure that some honest commitment $c_i$ in the initial list will end up in a position that is known to the adversary at the end of the shuffling protocol. To achieve this, the adversary merely needs to spend $T$ adaptive corruptions to corrupt the one shuffler in each round, who is supposed to touch commitment $c_i$. Thus, for such an approach to work, we either need a protocol with a large number of rounds or an adversary that can only adaptively corrupt very few of the shufflers.

In order to tolerate more adaptive corruptions, the protocol could somehow require each commitment to be shuffled by multiple parties in each round. This approach, however, only leads to a mediocre improvement in the number of adaptive corruptions that can be tolerated at the cost of significantly more shuffles that are being performed during the protocol execution. If every shuffler performs $\ell$ shuffles of $k$ commitments each in each round, then the total number of shuffles in the protocol is increased by a multiplicative factor of $\ell$. At the same time, the adversary's cost of performing the same attack as above is also only increased by the same factor $\ell$. The main problem with the approaches outlined above is that they always publicly reveal who will shuffle which subsets of commitments.

Somewhat surprisingly, it turns out that one can tolerate a substantially larger amount of adaptive corruptions, when the shuffling protocol does not publicly reveal which shuffler will shuffle which subset. The main idea here is that the adversary cannot efficiently prevent certain commitments from being shuffled, since it does not know who shuffles what. Not publicly coordinating who shuffles what in each round, however, leads to two new problems of its own. Firstly, how do we ensure that each commitment is being shuffled often enough? Secondly, what do we do if two shufflers read and shuffle intersecting subsets of commitments. Recall that we would like the final shuffled list to be a permutation of the initial list.

Our protocol resolves these issues roughly as follows. Assume we are in round $t$ of the protocol execution and $(c_1, \ldots, c_n)$ are the somewhat shuffled commitments from round $t-1$. At the beginning of the round, the randomness beacon $\mathcal{B}_\pi(t)$ returns a permutation $\pi^t$ over $[n]$, which effectively defines a partition of the $n$ commitments into $n/k$ batches. The commitments belonging to batch $i \in \{0, \ldots, n/k-1\}$ are defined as the commitments $\left(c_{\pi(i \cdot k+1)}, \ldots, c_{\pi(i \cdot k+k)}\right)$. Every shuffler $S_j$ for

$j \in [m]$ independently selects a batch, locally shuffles it, and publishes this shuffle on the bulletin board. At the end of the round each batch was shuffled either zero, one, or multiple times. In the next round, we use the randomness beacon to select one winning shuffle per batch that was shuffled at least once and all other shuffles for that batch will be discarded and ignored. Any untouched batch simply proceeds to the next round as is. If a batch in a round is not shuffled by anybody or if a adversarially corrupted shuffler is selected as the winning shuffle, then we say this batch failed. Any batch that does not fail, then succeeds. If the size $k$ of each local shuffle is not too small in the sense that the number of batches $n/k$ is not too large compared to the number $m - \beta$ of honest shufflers, then we are guaranteed that most commitments will be shuffled, and in particular, shuffled by somebody honest in each round. In other words, we are guaranteed that after some, as we will show, not too large number of rounds all honest commitments will have been shuffled by an honest party sufficiently often to hide their location in the final shuffled list.

**Formal Protocol Description.** Armed with the intuition from above, we are now ready to present our full protocol, which can be found in Figure 2. The main result of this paper in its full generality is stated Theorem 5, which can be found in Section 5 along with its proof. The full theorem statement is quite technical and can be challenging to parse. For this reason we first provide Corollaries 3 and 4 with simpler statements for some fixed choices of parameters.

The first corollary considers parameter ranges, where the number of shufflers $m$ is sufficiently large compared to the number of batches $n/k$ and it assumes that $k$ is not too small. Informally, the Corollary 3 tells us that the round complexity only depends logarithmically on the length of the vector that is being shuffled, even if a large fraction of the commitments are corrupted. The resulting shuffle will provide security with an overwhelming probability and the adversaries winning probability is only a constant factor better than pure guessing.

**Corollary 3.** *Let $n$ be the number of commitments and $m$ the number of parties. Assume at most a $\mu$-fraction of shufflers are corrupted in any round and that at most a $4/5$-fraction of commitments have been opened and at most a $4/5$-fraction of commitments are adversarially corrupted. Let* $(\mathtt{Commit}, \mathtt{Open})$ *be a perfectly hiding and computationally binding commitment scheme. Let* `Shuffle` *be a computationally hiding local shuffling scheme. Then there are universal constants $c_1, c_2, c_3, c_4 > 0$ such that for any $\delta \geq \exp(-o(n))$, if we have $T = c_1(\lg_{1/\mu}(n/\delta) + \lg_k n)$, $k \geq c_2 \lg^2(n/\delta)$ and $m \geq c_3(n/k)\lg(1/\mu)$, then it holds with probability at least $1 - \delta$ that*

$$\Pr\left[\begin{matrix}(z, \pi) \leftarrow \langle \Pi(r), \mathcal{A}_0(\tilde{r})\rangle \\ (i, j) \leftarrow \mathcal{A}_1^{\mathcal{O}(\pi,\cdot)}(z)\end{matrix} : \pi(i) = j \wedge i \notin Q_\alpha \cup Q_\gamma\right] \leq c_4/n + \mathsf{negl}(\lambda)\,,$$

Our next corollary considers the setting, where the number of parties $m$ is smaller than the number of batches $n/k$. Effectively, Corollary 4 tells us that the $m$ being smaller than $n/k$ by a factor of $\xi$ requires us to pick $k$ and $T$ by a factor of $\xi$ larger.

**Corollary 4.** *Let $n$ be the number of commitments and $m = \xi(n/k)$ the number of parties for some $0 < \xi < 1/2$. Assume at most a $4/5$-fraction of shufflers are corrupted in any round and that at most a $4/5$-fraction of commitments have been opened and at most a $4/5$-fraction of commitments are adversarially corrupted. Let* $(\mathtt{Commit}, \mathtt{Open})$ *be a perfectly hiding and computationally commitment scheme. Let* `Shuffle` *be a computationally hiding local shuffling scheme. Then there are universal*

$\texttt{DistributedShuffle}(c_1, \ldots, c_n)$

$(\tilde{c}_1, \ldots, \tilde{c}_n) := (c_1, \ldots, c_n)$
**for** $i \in [m]$ :
  Shuffler $S_i$ picks $(v_i^1, \ldots, v_i^T) \leftarrow \mathbb{Z}_{n/k}^T$
  $S_i$ publishes $(e_i^1, \ldots, e_i^T)$,
  where $(e_i^j, d_i^j) \leftarrow \texttt{Commit}(v_i^j)$ for $j \in [T]$
$(w_1^1, \ldots, w_m^T) \leftarrow \mathcal{B}_r(0)$
**for** $t \in [T+1]$ :
  $\pi^t \leftarrow \mathcal{B}_\pi(t)$ // Public permutation for round $t$
  **if** $t > 1$ : // Select one shuffle per batch from last round
    $\left(u_0, \ldots, u_{n/k-1}\right) \leftarrow \mathcal{B}_s\left(\ell_0, \ldots, \ell_{n/k-1}\right)$
    **for** $i \in \{0, \ldots, n/k-1\}$ **with** $\ell_i > 0$ :
      $(j_1, \ldots, j_k) := (\pi_{t-1}(i \cdot k + 1), \ldots, \pi_{t-1}(i \cdot k + k))$
      **parse** $q_i^{u_i}$ **as** $\left(d_j^{t-1}, j, s, (\hat{c}_{j_1}, \ldots, \hat{c}_{j_k})\right)$
      **if** $\texttt{Open}(e_j^{t-1}, d_j^{t-1}) + w_j^{t-1} = s$ : // Was correct batch shuffled?
        $(\tilde{c}_{j_1}, \ldots, \tilde{c}_{j_k}) := \left(\hat{c}_{j_1}^{b_j}, \ldots, \hat{c}_{j_k}^{b_j}\right)$
  **if** $t \in [T]$ : // Shuffling of batches
    $(\ell_0, \ldots, \ell_{n/k-1}) := (0, \ldots, 0)$ // Counter for each batch
    **for** $i \in [m]$ : // Each shuffler of round
      $s := \left(v_i^t + w_i^t \mod n/k\right)$ // Batch to be shuffled by $S_i$
      $(j_1, \ldots, j_k) := (\pi_t(s \cdot k + 1), \ldots, \pi_t(s \cdot k + k))$ // Batch $s$
      $S_i$ computes $(\hat{c}_{j_1}, \ldots, \hat{c}_{j_k}) \leftarrow \texttt{Shuffle}\left(\tilde{c}_{j_1}, \ldots, \tilde{c}_{j_k}\right)$
      $S_i$ publishes $q_s^{\ell_s} := \left(d_i^t, i, s, (\hat{c}_{j_1}, \ldots, \hat{c}_{j_k})\right)$
      $\ell_s := \ell_s + 1$ // Increase counter of batch that is shuffled
$(j_1, \ldots, j_n) := (\pi_{T+1}(1), \ldots, \pi_{T+1}(n))$ // One final public permutation
**return** $(\tilde{c}_{j_1}, \ldots, \tilde{c}_{j_n})$

**Fig. 2.** Formal description of the distributed shuffling protocol among $m$ parties for shuffling $n$ commitments with $k$-sized shuffles.

*constants* $c_1, c_2, c_3, c_4 > 0$ *such that for any* $\delta \geq \exp(-o(n))$*, if we have* $T = c_1 \xi^{-1} \lg(n/\delta)$ *and* $k \geq c_2(\lg^2(n/\delta) + \xi^{-1} \lg(n/\delta))$*, then it holds with probability at least* $1 - \delta$ *that*

$$\Pr\left[\begin{matrix}(z,\pi) \leftarrow \langle \Pi(r), \mathcal{A}_0(\tilde{r})\rangle \\ (i,j) \leftarrow \mathcal{A}_1^{\mathcal{O}(\pi,\cdot)}(z)\end{matrix} : \pi(i) = j \wedge i \notin Q_\alpha \cup Q_\gamma\right] \leq c_4/n + \mathsf{negl}(\lambda)\,,$$

It will not be directly obvious that Corollary 4 follows from Theorem 5 and for this reason we provide a proof thereof in Section 5.3.

## 5 Proof of Theorem 5

In this section, we proceed to prove our main theorem, which is stated below.

**Theorem 5.** *Let* $n$ *be the number of commitments,* $m$ *the number of parties,* $k$ *the size of a local shuffle. Let* $\alpha$ *be the maximum number of adversarially corrupted commitments, let* $\beta$ *be the maximum number of corrupted shufflers per round, and let* $\gamma$ *be the maximum number of opened commitments after the shuffle. Let* $(\mathtt{Commit}, \mathtt{Open})$ *be a perfectly hiding and computationally binding commitment scheme. Let* `Shuffle` *be a computationally hiding local shuffling scheme. Then the following statements are all true:*

*(i) The set of values in the output commitments is a permutation of values in the input commitments.*

*(ii) The probability that a batch fails is* $p$ *with*

$$p = (1 - k/n)^m + (1 - (1 - k/n)^m) \cdot \frac{\beta}{m}.$$

*(iii) Let* $T = 10 \lg_{1/p}(n/\delta) + 4 \lg_k(n)$ *and* $\eta = \gamma + \alpha - \gamma\alpha/n + \sqrt{\alpha \ln(1/\delta)}$*. Then for any* $k$ *with*

$$k \geq \max\left\{\begin{matrix}(64/9)^2 \ln^2(8n^3 \lg_2(n)/\delta) \left(\frac{n}{n-\eta}\right)^2, \\ \ln(8n^3 \lg_2(n)/\delta) \frac{8n}{n-\eta}, \\ 3e \lg_{1/p}(n/\delta) + e \lg_k n\end{matrix}\right\}$$

*it holds that with probability at least* $1 - \delta$ *that*

$$\Pr\left[\begin{matrix}(z,\pi) \leftarrow \langle \Pi(r), \mathcal{A}_0(\tilde{r})\rangle \\ (i,j) \leftarrow \mathcal{A}_1^{\mathcal{O}(\pi,\cdot)}(z)\end{matrix} : \pi(i) = j \wedge i \notin Q_\alpha \cup Q_\gamma\right] \leq 4/(n-\eta) + \mathsf{negl}(\lambda)\,,$$

*where the probability is taken over the random coins* $r$ *and* $\tilde{r}$ *as well as the random choice of the randomness beacon.*

*Proof.* To see that our scheme outputs a permutation of the input list, we make the following observations. In every round, every batch may be shuffled more than once, but in the successive round exactly one shuffle per batch survives. Furthermore, we recall that in our model, the adversary never lies about which batch it actually shuffled. Thus in every round, we have a valid permutation of the input list and therefore the protocol produces a correct output.

Let $\mathsf{hybrid}_0$ be the the protocol execution in the presence of an adversary as defined in Definition 6. Let us consider $\mathsf{hybrid}_1$, which is identical to $\mathsf{hybrid}_0$ with the only difference being that we replace all commitments in the input vector with commitments to 0. We observe that in $\mathsf{hybrid}_1$, due to the perfect re-randomizability of the input commitments, each local shuffle now also returns input and output vectors of length $k$ that perfectly hide the permutation between them. Due to the computationally hiding property of the commitments, we know that the adversary's success probability in $\mathsf{hybrid}_0$ and $\mathsf{hybrid}_1$ can at most differ by a negligible in $\lambda$ amount. In the following we will analyze the success probability of the adversary in $\mathsf{hybrid}_1$.

Furthermore, let us make the following observations. For each shuffler $S_i$, the adversary sees commitments $\left(e_i^1, \ldots, e_i^T\right)$ to the values $v_i^j \leftarrow \texttt{Commit}(v_i^j)$ for $j \in [T]$. Together with the output $\left(w_1^1, \ldots, w_m^T\right)$ of randomness beacon $\mathcal{B}_r(0)$, these values define which batches each party will shuffle throughout the protocol execution. By assumption, the commitment scheme used for these commitments is perfectly hiding, which means that the adversary obtains no information about the committed values of the honest parties. The best the adversary can do is guess which batch will be shuffled by an honest shuffler. Furthermore, we observe that in every round $t$, every adversarially corrupt party $S_i$ is forced to shuffle a uniformly random batch $\left(v_i^t + w_i^t \mod n/k\right)$, unless it were to break the computational binding property of the commitment scheme.

Let us now consider our starting vector $c_1, \ldots, c_n$ of commitments and fix some index $z \in [n]$, which is not adversarially corrupted and whose location in the final permuted vector is not yet revealed. Let us now see how well this index will be hidden by the shuffling protocol. To obtain our final statement we will simply perform a union bound over all $z \in [n]$ that are honest and not yet opened.

We will view each commitment as a cup of water. Initially cup $z$ has 1 unit of water and the remaining have 0. This can be interpreted as the adversary knowing with certainty where the $z$-th commitment is before the shuffling starts. Among the cups, $\eta$ of them are *idle* and the remaining $n - \eta$ cups are *active*. The idle commitments correspond to those either controlled by the adversary or already opened at the end of the protocol execution. The active ones are those belonging to an honest shuffler that are not yet revealed. For now, we deal with the case of a fixed $\eta$ and in Section 5.2, we generalize the result to the whole execution of opening up to $\gamma$ commits, including also the possibility that adversarial commits have been opened.

For $T$ rounds, the cups are randomly shuffled and partitioned into $n/k$ *batches* of $k$ cups each. Let $B_1^t, \ldots, B_{n/k}^t$ denote the batches in round $t$.

In each round, there are $\beta$ *adversarial* tokens and $m - \beta$ *honest* tokens. The tokens are each assigned to a uniform random batch and among the tokens assigned to a batch, a uniform random one is chosen. If a batch either chooses an adversarial token or is not assigned any token, then the batch *fails*. The batches that do not fail *succeed*. The assignment of tokens correspond to the protocol in the sense that a batch is either assigned an honest shuffler, an adversarial shuffler or no shuffler at all.

In each succeeding batch $B_i^t$, all water from active cups is collected and distributed evenly among the active cups. This intuitively corresponds to the fact that the local shuffling operation distributes the probability of a commitment being in a certain position evenly among all other honest commitments within that shuffle. We use $C_i^t \subseteq B_i^t$ to denote the active cups in batch $B_i^t$. We also define $b_{t,i}$ as the amount of water in the $i$'th cup after round $t$. The initial water in the cups are denoted by $b_{0,i}$. Finally, we define $W(B_i^t) = \sum_{j \in C_i^t} b_{t-1,j} / |C_i^t|$ as the average amount of

water in the active cups entering batch $i$ during round $t$. We remark that for all $j \in C_i^t$, we have $b_{t,j} = W(B_i^t)$.

Let us also bound the probability that a batch fails. A batch fails if it is not assigned an honest token. The probability that a batch receives any token is $1-(1-k/n)^m$. Conditioned on receiving a token, it receives an adversarial token with probability $\beta/m$. Hence it fails with probability $p$ satisfying:

$$p = (1-k/n)^m + (1-(1-k/n)^m) \cdot \frac{\beta}{m}.$$

We wish to bound the probability that after $T$ rounds, there is a cup with more than $4/(n-\eta)$ units of water in it. To do so, we identify a few different things that may go wrong. Intuitively, there are two different ways that we could have a cup of water with more than $4/(n-\eta)$ units of water in it after each round. Either (1), many cups with slightly more than $4/(n-\eta)$ units of water appear in the same batch $B_i^t$, or (2) a cup with much more than $4/(n-\eta)$ units of water appeared in a batch. We claim that it is very unlikely that (1) ever occurs and we claim that (2) cannot occur too often. For (1), we show:

**Lemma 6.** *If we have*

$$k \geq \max \left\{ \begin{array}{c} (64/9)^2 \ln^2(8n^3 \lg_2(n)/\delta) \left(\frac{n}{n-\eta}\right)^2, \\ \ln(8n^3 \lg_2(n)/\delta) \frac{8n}{n-\eta} \end{array} \right\}$$

*and $T \leq 4k/e$, then*

$$\Pr\left[\exists t, i : \max_{j \in C_i^t} b_{t-1,j} \leq k^{1/4} W(B_i^t) \wedge W(B_i^t) \geq 4/(n-\eta)\right] \leq \delta/2.$$

Lemma 6 shows that it is very unlikely that many cups without a lot of water combine to create a batch with above $4/(n-\eta)$ units of water. Intuitively, this is true since without cups having a lot water, we get strong concentration on the average amount of water in a batch. We defer the proof of the lemma to Section 5.1.

Despite ruling out many cups without a lot of water creating a batch with above $4/(n-\eta)$ water, there may still be a few cups with lots of water. In particular, there is a single cup with 1 unit of water in the beginning. For step (2), we argue about such cups by considering a notion of a witness:

*Witness.* We define a *witness* of failure. Such a witness consists of up to $a$ indices $1 \leq i_1 < i_2 \cdots < i_a \leq T$ and for each index $i_j$, also an integer $x_j \in [k]$. Here $a$ is a parameter to be fixed. Note that there are no more than

$$\sum_{i=0}^{a} \binom{T}{i} k^i \leq (eTk/a)^a$$

distinct witnesses. A witness $W = (i_1, \ldots, i_a, x_1, \ldots, x_a)$ and an execution of the shuffling produces a *trace* as follows: Before round 1, let $h_0 = 1$ be an index pointing to the batch with all the water. Then process the rounds in turn, from $t = 1, \ldots, T$. When processing round $t$, if $t \neq i_j$ for any $j$, we simply let $h_t = h_{t-1}$. Otherwise, we update $h_t$ to equal the $x_j$'th cup from the batch containing $h_{t-1}$ in round $t$. The trace of a witness is thus a sequence of cups that change only in rounds $i_j$.

The trace $h_0, \ldots, h_T$ of a witness $W = (i_1, \ldots, i_a, x_1, \ldots, x_a)$ is *valid* if:

– For all $t \in \{i_1, \ldots, i_a\}$, it holds that $b_{t-1,h_{t-1}} \geq k^{1/4} \cdot b_{t,h_t}$.
– For all $t \notin \{i_1, \ldots, i_a\}$, it holds that $h_t$ is in a failing batch in round $t$.
– $b_{T,h_T} \geq 4/(n-\eta)$.

These properties already puts some constraints on the number of valid witnesses. Concretely, observe that for any $t \in \{i_1, \ldots, i_a\}$, the amount of water in the cup $b_{t-1,h_{t-1}}$ increases by a factor at least $k^{1/4}$ over $b_{t,h_t}$. Since we end with $b_{T,h_T} \geq 4/(n-\eta)$ and start with $b_{0,h_0} = 1$, it follows that $a \leq 4 \lg_k(n-\eta)$. We will thus only consider witnesses with $a \leq 4 \lg_k n$.

To build some intuition for a witness and its trace, note that it if we look at the path from the last round and back to the first, it corresponds to tracing a path backwards through the rounds of shuffling, all the time keeping track of a cup with a lot of water. In each round, either the traced cup is in a failing batch, or one of the cups entering the batch had much more water in the previous round (a factor $k^{1/4}$). We then trace that cup with lots of water. Since the amount of water in the traced cup increases a lot (recall that we trace a cup backwards through the rounds) whenever we are not in a failing batch, this cannot happen too often.

The crucial point is that if we rule out the unlikely event defined in Lemma 6, then there can only be a cup with more than $4/(n-\eta)$ water left at the end, if there is a witness with a valid trace:

**Lemma 7.** *If for all $t, i$, it holds that $\max_{j \in C_i^t} b_{t-1,j} \geq k^{1/4} W(B_i^t)$ or $W(B_i^t) < 4/(n-\eta)$, then if there is a cup $j$ with $b_{T,j} \geq 4/(n-\eta)$, then there must be a witness with a valid trace.*

*Proof.* Assume that for all $t, i$, it holds that $\max_{j \in S_i^t} b_{t-1,j} \geq k^{1/4} W(B_i^t)$ or $W(B_i^t) < 4/(n-\eta)$, but that there is an $j$ such that $b_{T,j} \geq 4/(n-\eta)$. In this case, we claim there must be a witness with a valid trace.

We find the witness and trace starting from the last round and trace it backwards. That is, we start by setting $h_T = j$ for an index $j$ with $b_{T,j} \geq 4/(n-\eta)$. Then we trace a cup back through the rounds. Assume we are at round $t+1$ and want to determine $h_t$. Consider the batch that contains the cup $h_{t+1}$ in round $t$. The average amount of water among the active cups in that batch is precisely $b_{t+1,h_{t+1}}$. If the batch failed in round $t$, we simply set $h_t = h_{t+1}$ and observe that $b_{t,h_t} = b_{t+1,h_{t+1}}$. Otherwise, the batch did not fail and we let $h_t$ be the index of the cup with the most water entering the same batch as $h_{t+1}$ in round $t$. Clearly, $b_{t,h_t} \geq b_{t+1,h_{t+1}}$ since we pick the max. Thus the values $b_{t,h_t}$ may only increase as we go back and thus stay above $4/(n-\eta)$. Thus using the assumption that $\max_{j \in C_i^t} b_{t-1,j} \geq k^{1/4} W(B_i^t)$ or $W(B_i^t) < 4/(n-\eta)$ for all $i, t$, we must in fact have $b_{t,h_t} > k^{1/4} b_{t+1,h_{t+1}}$. It follows that the trace we construct is indeed valid.

In light of Lemma 7, what remains is to argue that it is unlikely to have a witness with a valid trace:

**Lemma 8.** *Let $p$ be the probability that a batch fails. If we have $k \geq 3e \lg_{1/p}(n/\delta) + e \lg_k n$ and set $T = 10 \lg_{1/p}(n/\delta) + 4 \lg_k n$, then $T \leq 4k/e$ and the probability that there exists a witness with a valid trace is at most $\delta^{10}/n^2$.*

*Proof.* Observe that for a fixed witness $W$, the probability that its trace is valid is at most $p^{T-a}$, where $p$ is the probability that a fixed batch fails in a round. If we let $T$ satisfy

$$\left(\frac{eTk}{4 \lg_k n}\right)^{4 \lg_k n} \cdot p^{T - 4 \lg_k n} \leq \delta^{10}/n^2,$$

then a union bound over all possible witnesses shows that the probability that there is witness with a valid trace is no more than $\delta^{10}/n^2$. If we assume $k \geq 3\lg_{1/p}(n/\delta) + e\lg_k n$ and set $T = 10\lg_{1/p}(n/\delta) + 4\lg_k n$, then $T \leq 4k/e$ and we see that the above is satisfied:

$$\begin{aligned}&\left(\frac{eTk}{4\lg_k n}\right)^{4\lg_k n} \cdot p^{T-4\lg_k n}\\ \leq &\left(k^2\right)^{4\lg_k n} \cdot p^{10\lg_{1/p}(n/\delta)} = n^8(\delta/n)^{10} \leq \delta^{10}/n^2.\end{aligned}$$

**Corollary 9.** *Let $p$ be the probability that a batch fails. If we set $T = 10\lg_{1/p}(n/\delta) + 4\lg_k n$ and have*

$$k \geq \max\left\{\begin{array}{c}(64/9)^2\ln^2(8n^3\lg_2(n)/\delta)\left(\frac{n}{n-\eta}\right)^2,\\ \ln(8n^3\lg_2(n)/\delta)\frac{8n}{n-\eta},\\ 3e\lg_{1/p}(n/\delta) + e\lg_k n\end{array}\right\},$$

*then with probability at least $1 - \delta/(2n^2) - \delta^{10}/n^2$, there is no index $j$ with $b_{T,j} \geq 4/(n-\eta)$.*

*Proof.* By Lemma 8, we have $T \geq 4k/e$ and by the requirement on $k$, Lemma 6 gives us that

$$\begin{aligned}&\Pr\left[\exists t, i : \max_{j\in C_i^t} b_{t-1,j} \leq k^{1/4}W(B_i^t) \wedge W(B_i^t) \geq 4/(n-\eta)\right]\\ \leq &\delta/(2n^2).\end{aligned}$$

Combining this with Lemma 8 gives us that with probability at least $1-\delta/(2n^2)-\delta^{10}/n^2$, there is no witness with a valid trace and for all $i, t$ we have $\max_{j\in C_i^t} b_{t-1,j} \geq k^{1/4}W(B_i^t)$ or $W(B_i^t) < 4/(n-\eta)$. Lemma 7 finally implies that there is no cup $j$ with $b_{T,j} \geq 4/(n-r)$.

### 5.1 Proof of Lemma 6

For group $i$ in round $t$, denote by $E_{t,i}$ the undesirable event that $\max_{j\in C_i^t} b_{t-1,j} \leq k^{1/4}W(B_i^t) \wedge W(B_i^t) \geq 4/(n-\eta)$. To bound $\Pr[E_{t,i}]$, we further define events $E_{t,i,v}$ that occur if $v \leq W(B_i^t) \leq 2v$ and $\max_{j\in C_i^t} b_{t-1,j} \leq 2k^{1/4}v$. Then $\Pr[E_{t,i}] \leq \sum_{h=0}^{\lg_2(n-\eta)-2} \Pr[E_{t,i,2^{h+2}/(n-\eta)}]$.

To bound $\Pr[E_{t,i,2^{h+2}/(n-\eta)}]$, let $v = 2^{h+2}/(n-\eta)$ and observe that:

$$\begin{aligned}\Pr[E_{t,i,v}] &= \sum_{s=0}^{k} \Pr[|C_i^t| = s]\Pr[E_{t,i,v} \mid |C_i^t| = s]\\ &= \sum_{s=0}^{k} \Pr[|C_i^t| = s]\Pr[\max_{j\in C_i^t} b_{t-1,j} \leq k^{1/4}2v \mid |C_i^t| = s]\\ &\quad \cdot \Pr[E_{t,i,v} \mid \max_{j\in C_i^t} b_{t-1,j} \leq k^{1/4}2v, |C_i^t| = s]\\ &\leq \sum_{s=0}^{k} \Pr[|C_i^t| = s]\cdot\\ &\quad \Pr[W(B_i^t) \geq v \mid \max_{j\in C_i^t} b_{t-1,j} \leq k^{1/4}2v, |C_i^t| = s].\end{aligned}$$

Conditioned on $\max_{j \in S_i^t} b_{t-1,j} \leq k^{1/4}2v, |C_i^t| = s$, the cups in $C_i^t$ are distributed precisely as $s$ samples without replacement from the set of all active cups $h$ with $b_{t-1,h} \leq k^{1/4}2v$. If we consider one such sample/cup, then the expected amount of water in it is at most that of a sample from among all active indices (since removing the cups with the most water only decreases the expectation). Thus each of the $s$ samples contains no more than $1/(n-\eta) \leq v/4$ water in expectation. It follows from a Hoeffding bounding without replacement that

$$\begin{aligned}&\Pr[W(B_i^t) \geq v \mid \max_{j \in C_i^t} b_{t-1,j} \leq k^{1/4}2v, |C_i^t| = s] \\ &\leq \exp\left(-\frac{2(3/4)^2 v^2 s^2}{s 4 k^{1/2} v^2}\right) = \exp\left(-\frac{9s}{32\sqrt{k}}\right).\end{aligned}$$

Notice that $\mathbb{E}[|C_i^t|] = k \cdot \frac{n-\eta}{n}$. By Chernoff, we have

$$\Pr\left[|C_i^t| < \frac{k(n-\eta)}{2n}\right] \leq \exp\left(-k \cdot \frac{n-\eta}{8n}\right)$$

If we assume $k \geq \ln(8n^3 \lg_2 n/\delta)\frac{8n}{n-\eta}$, then this is no more than $\delta/(8n^3 \lg_2 n)$. Hence

$$\Pr[E_{t,i,v}] \leq \delta/(8n^3 \lg_2 n) + \exp\left(-\frac{9\sqrt{k}(n-\eta)}{64n}\right).$$

Let us also assume $k \geq (64/9)^2 \ln^2(8n^3 \lg_2(n)/\delta)\left(\frac{n}{n-\eta}\right)^2$, then this is at most $\delta/(4n^3 \lg_2 n)$. A union bound over all $T$ rounds, all $n/k$ groups and all $\lg_2(n-\eta) - 2 \leq \lg_2 n$ values of $v$ shows that the probability that there is an $E_{t,i}$ that occurs is at most (using $T \leq 4k/e$):

$$(n/k)(4k/e)(\delta \lg_2 n)/(4n^3 \lg_2 n) = \delta/(n^2 e) < \delta/(2n^2).$$

### 5.2 Random Number of Idle Cups.

In the previous analysis, we assumed that a fixed number of cups, denoted $\eta$, were idle. We will now generalize the results to the following setup: Before the random shuffling process begins, we have two phases. In the first phase, we have a fixed set of $\alpha$ marked cups. In the second phase, we choose a uniform random subset of $\gamma$ of the cups and mark them. If a cup was marked either during the first or second phase, it becomes idle and otherwise it is active. Notice that this corresponds to first having chosen the $\alpha$ adversarially corrupted commitments and then opening $\gamma$ random commits.

Once the idle and active cups have been chosen, we run the water shuffling process from above. We now bound the probability of seeing an active cup with more than $4/(n-\eta)$ units of water after $T$ rounds. As the guarantees of the previous section goes down with the number of idle cups $\eta$, we first bound the probability of seeing many idle cups. For this, notice that the first phase marks precisely $\alpha$ cups. For the second phase, the number of newly marked cups can be bounded by observing that the $\gamma$ samples without replacement each picks a cup already marked in the first phase with probability precisely $\alpha/n$ (when looking at the marginal distribution of the cup). It follows by a Hoeffding bound without replacement that the number of newly marked cups in the second phase, denoted $y$, satisfies:

$$\Pr[y - (1-\alpha/n)\gamma > t] < \exp(-2t^2/\gamma).$$

| # | $n$ | $m$ | $k$ | $\alpha/n$ | $\beta/m$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $2^{14}$ | $2^{14}$ | 128 | 1/16 | 1/16 | $T$ | 3 | 4 | 5 | 6 | 7 | 8 | | | |
| | | | | | | $\delta$ | 0.8500 | 0.1800 | 0.02215 | 0.00220 | 0.0002 | 0.0000 | | | |
| 2 | $2^{14}$ | $2^{14}$ | 128 | 1/4 | 1/4 | $T$ | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| | | | | | | $\delta$ | 0.4656 | 0.1910 | 0.0677 | 0.0219 | 0.0066 | 0.0018 | 0.0006 | 0.0002 | 0.0000 |
| 3 | $2^{14}$ | 128 | 128 | 1/16 | 1/16 | $T$ | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 |
| | | | | | | $\delta$ | 0.3357 | 0.1384 | 0.0523 | 0.0191 | 0.0061 | 0.0017 | 0.0006 | 0.0005 | 0.0001 |
| 4 | $2^{14}$ | 16 | $2^{10}$ | 1/16 | 1/16 | $T$ | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | |
| | | | | | | $\delta$ | 0.5194 | 0.1346 | 0.0362 | 0.0087 | 0.0023 | 0.0007 | 0.0002 | 0.0000 | |
| 5 | $2^{14}$ | 16 | 128 | 1/16 | 1/16 | $T$ | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | |
| | | | | | | $\delta$ | 0.9958 | 0.6230 | 0.1558 | 0.0264 | 0.0026 | 0.004 | 0.0002 | 0.0000 | |

**Table 1.** Results of our numerical experiments for determining the number of rounds $T$ that is needed for successfully shuffling with different sets of parameters.

Setting $t = \sqrt{\gamma \ln(1/(n\delta))}$ bounds the above by $\delta^2/n^2$. Thus with probability at least $1 - \delta^2/n^2$, we have $\eta <= \alpha + y <= \alpha + t + (1 - \alpha/n)\gamma = \alpha + \gamma - \alpha\gamma/n + \sqrt{\gamma \ln(1/\delta)}$. A union bound together with Corollary 9 gives (assuming $\delta$ is small enough that $1 - \delta/(2n^2) - \delta^{10}/n^2 - \delta^2/n^2 \geq 1 - \delta/n^2$) us that with probability at least $1 - \delta/n^2$, there is no index $j$ with $b_{T,j} \geq 4/(n - \eta)$. The above analysis was for a fixed number of opened commits $\gamma$ and a fixed input cup $z \in [n]$. Doing a union bound over all pairs of a $\gamma' \leq \gamma$ and all $z \in [n]$ that the probability that throughout the opening of $\gamma$ commits, that there is ever an input cup $z$ whose output destination can be predicted with probability greater than $4/(n-\eta)$ is at most $n^2 \cdot (\delta/n^2) = \delta$. This completes the proof of Theorem 5.

### 5.3 Proof of Corollary 4

*Proof.* We see that the failure probability of a batch, $p$, is no more than $(1 - k/n)^m + (1 - (1 - k/n)^m)(4/5)$. For any $x$ satisfying $(1 - k/n)^m \leq x < 1$, this is upper bounded by $x + (1 - x)(4/5)$. We use that $(1-k/n)^m \leq \exp(-km/n) = \exp(-\xi) \leq \exp(-(\xi/2)\sum_{n=1}^{\infty}(\xi/2)^{n-1}/n) = 1 - \xi/2$ by a Taylor series for $\ln(1 - x)$ and using $\sum_{n=0}^{\infty}(\xi/2)^n/n < 2$. Thus we conclude $p = (1 - k/n)^m + (1 - (1-k/n)^m)(4/5) \leq (1-\xi/2)+(\xi/2)(4/5) \leq 1-\xi/2$. Hence $\lg_{1/p}(n/\delta) = \ln(n/\delta)/\ln((1-\xi/2)^{-1}) \leq \ln(n/\delta)/\ln(\exp(-\xi/2)^{-1}) = (2/\xi)\ln(n/\delta)$. Plugging this into Theorem 5 concludes the proof.

## 6 Experiments

In this section, we perform numerical experiments to precisely determine the practical constants in our distributed shuffling protocol. In our experiments we consider $n$ commitments and $m$ shufflers of which $\beta$ are adversarial. Moreover, we assume that $\eta = (3/4)n$ commitments have already been opened and that $\alpha$ commitments are adversarially corrupted. We then repeatedly simulate the water mixing process from the proof of Theorem 5 with $T$ rounds and check whether the next commitment to be opened can be determined with probability greater than $4/(n - \eta) = 16/n$, i.e. if there is an input commitment that has more than $4/(n - \eta)$ water for the next position to be opened. If so, we count the simulation as failing and if not, we count it as succeeding. In this way, the fraction of failing simulations, denoted $\delta$, is an unbiased estimate of the true probability that the next commitment to be opened can be determined with probability greater than $4/(n-\eta)$. We run the experiment with varying values of $T$ and $k$. The number of simulations we run for each

experiment is 20000. From our theoretical results, we know that $T = O(\lg_{1/p}(n/\delta) + \lg_k n)$ rounds suffice and thus we expect that the failure probability decreases by a constant factor, namely a factor $1/p$, with each round once the first few rounds have been performed.

Our benchmarks can be found in Table 1. In parameter set #1, we consider many shufflers performing small local shuffles and it can be seen that the shuffling terminates quickly, since the failure probability drops by a factor of $\approx 10$ in every round. In #2, we consider the same parameters with a larger set of corrupted commitments and shufflers, which results in a slightly larger round complexity to reach the desired failure probability. In every round, $\delta$ drops by around a factor of 3. In #3, we consider a small number of shufflers each performing small local shuffles. As expected it noticeably increases the round complexity, but even then the failure probability drops by a factor of 3 every two rounds. In #4, we consider an even smaller number of shufflers $m = 16$, but this time consider each of them performing a local shuffle of size $k = 2^{10}$ to shuffle a vector of length $2^{14}$. Our benchmarks show that this shuffle already terminates after 16 rounds. Lastly, in #5 we consider a extreme setting of parameters, where we have very few shufflers $m = 16$, which perform small shuffles of size $k = 128$. Our benchmarks show that for such parameters, the round complexity increases by an order of magnitude.

Concluding, our benchmarks show that for reasonable parameters, where we either have enough shufflers or where $k$ is chosen large enough, our distributed shuffling protocol terminates in a concretely small number of rounds.

## 7 Applications

In this section, we will highlight some of the applications that can benefit from our distributed shuffling protocol. We will focus on canonical examples in each of these applications, as providing fully fledged out solutions is beyond the scope of this work.

### 7.1 Single Secret Leader Elections

In the single secret leader election (SSLE) problem, recently introduced by Boneh et al. [BEHG20], we have $m$ parties with access to a public bulletin board that would like to elect exactly one leader among them. The leader should be fairly chosen, in the sense that each party should have a roughly equal probability of becoming the leader. Additionally, the leader should remain hidden until they decide to reveal themselves.

Boneh et al. present three solutions to this problem. The first two solutions are based on indistinguishability obfuscation and threshold fully homomorphic encryption respectively. Both of these tools are highly complex and computationally very expensive, thus these two solutions can be seen as theoretical feasibility results, rather than practical solutions ready for deployment today. The authors also outline a third solution based on a distributed shuffling protocol.[6] Here each of the $m$ participant $S_i$ publishes a commitment $c_i$ that only they can open on the bulletin board. Thereafter, all participants act as shufflers and permute the $n := m$ entries on the bulletin board using Håstad's square shuffle [Hås06, Hås16], which shuffles $n$ items using local shuffles of size $k = \sqrt{n}$. As mentioned in the introduction, Håstad's square shuffle is an algorithm whose

[6] SSLE based on shuffling has recently sparked some interested in the Ethereum ecosystem [Eth]. The protocol that is proposed there, however, has no well-defined adversarial model, no security proof, and is not secure against the adaptive corruption setting we consider in this work.

correctness is proven in a benign setting. It does not provide any provable security guarantees in the presence of an adversary that may aim to disrupt the shuffling process. In distributed systems such as cryptocurrencies, it may not always be realistic to assume that all shufflers are always online and that they all behave honestly.

Our distributed shuffling protocol allows for for solving the SSLE problem in a more realistic setting, where some participants can be either offline[7] or outright malicious. More generally, our protocol also allows for electing not just one, but an ordered list of up to $\gamma$ leaders. We note that our protocol is particularly well-suited for the SSLE setting, since here we naturally have more shufflers than batches. In such settings, our shuffling protocol is very efficient and only requires a small number of rounds to terminate. As a concrete example, one can see in Experiment #2 in Table 1 that our shuffling protocol among $m = n = 2^{14}$ shufflers with $k = 128$ is likely to terminate after only 13 rounds, even when 1/4 of all participants are malicious.

### 7.2 Mix-Nets

One of the most prominent and versatile tools for achieving anonymity in a variety of settings are mix-nets, which were introduced by Chaum [Cha81] already 40 years ago. For the sake of concreteness let us focus on the following exemplary applications. We have $n$ senders, who would each like to send one message to some set of receivers without revealing who sent a message to which receiver. Each sender sends their message to shuffler $S_1$, which shuffles all of them and forwards the resulting vector to $S_2$, who shuffles them again and so on and so forth. If we have $m$ shufflers and only one of them is honest, then the adversary has no ability to link any honest entry in the input vector to an entry in the output vector. The protocol has a round complexity that is linear in $m$. Apart from the large round-complexity, this approach also requires every single shuffler to have a large memory and to perform an amount of work that is linear in the length of the input vector. This is problematic for two reasons. From a scalability perspective, the length of the vector can quickly become a bottleneck hindering a large-scale usage of such mix-nets. From a participation perspective, the large computational and bandwidth costs for shufflers can be an entry barrier that prevents users to participate with their own machine in a distributed mix-net. Especially for distributed systems that aim to provide security guarantees like censorship resistance, it is important that parties with "normal" hardware can participate and that mix-nets do not depend on a few dedicated entities running it. Our distributed shuffling protocol provides a trade-off for classical mix-nets. If we assume that only some constant fraction of shufflers is adversarially corrupt, then our protocol shuffles all $n$ messages in $\mathcal{O}(\lg n)$ rounds. The total memory required by any shuffler is linear in $k$ and for certain parameter choices the total work of any shuffler can be sub-linear in the length $n$ of the input vector. Additionally, using our protocol as a mix-net provides security against sleeper attacks [Syv11], where an adversary may corrupt up to $\alpha$ of the senders to learn something about the locations of the honest senders' messages.

### 7.3 Electronic voting

One popular application of mix-nets is electronic voting [SK95, Nef01, JJR02]. In this setting we have $n$ citizens that would like to vote in an election. Glossing over many details, we have $m$ shufflers and we have a decryption committee that holds a decryption key $\mathsf{dk}$ corresponding to an encryption

[7] Offline shufflers can be seen as malicious shufflers that perform the identity permutation.

key ek. Each citizen casts its vote and encrypts it using ek. All encrypted votes are published on a bulletin board. The shufflers, then proceed to permute the votes before the decryption committee uses dk to decrypt every vote in the output vector. Given the decrypted vector anybody can tally the votes to determine the election outcome. As in the case of general mix-nets, our approach allows for reducing the memory and potentially also work required by each shuffler and allows for shuffling the input vector in only $\mathcal{O}(\lg n)$ rounds. Anonymity of an honest citizen's vote is maintained even if $\alpha$ citizens are maliciously corrupted or coerced.

## References


BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018. 3

BD92. Dave Bayer and Persi Diaconis. Trailing the dovetail shuffle to its lair. *The Annals of Applied Probability*, pages 294–313, 1992. 1.2

BEHG20. Dan Boneh, Saba Eskandarian, Lucjan Hanzlik, and Nicola Greco. Single secret leader election. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 12–24, 2020. 7.1

BG12. Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 263–280. Springer, 2012. 1, 3

BMG+07. Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against tor. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 11–20, 2007. 1

BNM+14. Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A Kroll, and Edward W Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *International Conference on Financial Cryptography and Data Security*, pages 486–504. Springer, 2014. 1

Cha81. David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981. 1, 7.2

DMS04. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004. 1

Eth. Ethereum. Whisk: A practical shuffle-based ssle protocol for ethereum. 6

FS01. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *Annual International Cryptology Conference*, pages 368–387. Springer, 2001. 1

GRS96. David M Goldschlag, Michael G Reed, and Paul F Syverson. Hiding routing information. In *International workshop on information hiding*, pages 137–150. Springer, 1996. 1

Hås06. Johan Håstad. The square lattice shuffle. *Random Structures and Algorithms*, 29(4):466–474, 2006. 1.2, 7.1

Hås16. Johan Håstad. The square lattice shuffle, correction. *Random Structures and Algorithms*, 48(1):213, 2016. 1.2, 7.1

HKR19. Max Hoffmann, Michael Klooß, and Andy Rupp. Efficient zero-knowledge arguments in the discrete log setting, revisited. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2093–2110, 2019. 3

JJR02. Markus Jakobsson, Ari Juels, and Ronald L Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *11th USENIX Security Symposium (USENIX Security 02)*, 2002. 1, 7.3

Max13. Coinjoin: Bitcoin privacy for the real world. 2013. 1

MD05. Steven J Murdoch and George Danezis. Low-cost traffic analysis of tor. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pages 183–195. IEEE, 2005. 1

MR14. Ben Morris and Phillip Rogaway. Sometimes-recurse shuffle. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 311–326. Springer, 2014. 1.2

MZ07. Steven J Murdoch and Piotr Zieliński. Sampled traffic analysis by internet-exchange-level adversaries. In *International workshop on privacy enhancing technologies*, pages 167–183. Springer, 2007. 1

Nef01. C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125, 2001. 1, 7.3

Ped91. Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991. 3

RSG98. Michael G Reed, Paul F Syverson, and David M Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected areas in Communications*, 16(4):482–494, 1998. 1

RY13. Thomas Ristenpart and Scott Yilek. The mix-and-cut shuffle: small-domain encryption secure against n queries. In *Annual Cryptology Conference*, pages 392–409. Springer, 2013. 1.2

SK95. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 393–403. Springer, 1995. 1, 7.3

Syv11. Paul Syverson. Sleeping dogs lie in a bed of onions but wake when mixed. *4th Hot Topics in Privacy Enhancing Technologies (HotPETs 2011)*, 2011. 7.2

Tho73. Edward O Thorp. Nonrandom shuffling with applications to the game of faro. *Journal of the American Statistical Association*, 68(344):842–847, 1973. 1.2