Distributed Shuffling in Adversarial Environments

Kasper Green Larsen
Aarhus University, Denmark

Maciej Obremski
National University of Singapore, Singapore

Mark Simkin
Ethereum Foundation, Aarhus, Denmark

Abstract

We study mix-nets in the context of cryptocurrencies. Here we have many computationally weak shufflers that speak one after another and want to joinlty shuffle a list of ciphertexts $(c_1, \ldots, c_n)$. Each shuffler can only permute $k << n$ ciphertexts at a time. An adversary $\mathcal{A}$ can track some of the ciphertexts and adaptively corrupt some of the shufflers.

We present a simple protocol for shuffling the list of ciphertexts efficiently. The main technical contribution of this work is to prove that our simple shuffling strategy does indeed provide good anonymity guarantees and at the same time terminates quickly.

Our shuffling algorithm provides a strict improvement over the current shuffling strategy in Ethereum's block proposer elections. Our algorithm is secure against a stronger adversary, provides provable security guarantees, and is comparably in efficiency to the current approach.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Distributed Computing, Shuffling

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.10

Related Version *Full Version*: `https://eprint.iacr.org/2022/560`

Funding *Kasper Green Larsen*: Supported by Independent Research Fund Denmark (DFF) Sapere Aude Research Leader grant No 9064-00068B.
Maciej Obremski: Funded by MOE2019-T2-1-145 Foundations of quantum-safe cryptography.

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 most applications, the vector entries are either commitments or ciphertexts and each position in the vector is associated with a corresponding identity. The process of shuffling the vector produces a new vector that contains the same multi-set of committed or encrypted values, but hides which value is associated to which identity. In anonymous communication systems [11, 26, 20], for instance, a set of senders would each like to communicate one message to a set of receivers without revealing who is talking to who. In electronic voting [26, 20, 23], 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 [21, 6], we have multiple payers, who would like to transfer money to multiple payees without revealing who is paying who.

A popular approach for achieving anonymity in the above applications are mix-nets [11]. Here, we assume the existence of one or more shufflers that shuffle the input vector one after another. If only one shuffler was honest, then even an adversary that corrupts all other shufflers cannot tell which entry in the input vector belongs to which entry in the output vector. From a security perspective this approach is great, but unfortunately such strong

anonymity guarantees do not come for free. The required memory and the computational overhead of each shuffler grows linearly in the length of the 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 memory burden on each shuffler. In addition, shufflers often need to provide computationally expensive zero-knowledge proofs attesting the correctness of their performed shuffle [26, 15, 23, 2] to show that no values in the vector have been changed by them. These high costs make mix-nets unsuitable for applications, where shuffling needs to terminate in a timely fashion and where the shufflers are restricted in terms of memory or computational power.

1.1 Our Contribution

In this work, we study mix-nets in the context of cryptocurrencies. Here we have many shufflers, but all of them are computationally weak, in the sense that they can only read and shuffle k entries in a vector of length n, where k is potentially much smaller than n. Initially, a vector of ciphertexts $(c_1, \ldots, c_n)$ is written on a public bulletin board, accessible to all. The shufflers speak one after another and each shuffler chooses k entries, re-randomizes, and permutes them. We assume that shuffling takes place in the presence of an adversary $\mathcal{A}$. At the start of the protocol, $\mathcal{A}$ is allowed to corrupt a subset of indices $I \subset \{1, \ldots, n\}$ with $|I| \leq \alpha$ and can track all ciphertexts c_i for $i \in I$ throughout the shuffling process. Additionally, the adversary can adaptively corrupt up to β shufflers throughout the execution. The goal of the shuffling protocol is to hide the output location of the uncorrupted entries in the input vector from the adversary. In terms of efficiency, we would like to minimize the number of shuffles of size k that need to be performed.

We present a very simple shuffling mechanism, where each shuffler picks k uniformly random entries and permutes them. The main technical contribution of this work is a upper bound that shows that this shuffling process terminates quickly and provides good anonymity guarantees. The following informal theorem is a corollary of our main theorem.

▶ **Theorem 1** (Informal). *Let $(c_1, \ldots, c_n)$ be a vector of ciphertexts. Let $\mathcal{A}$ be a PPT adversary that tracks $\alpha = C \cdot n$ ciphertexts, where C is a constant, and adaptively corrupts β shufflers adaptively. If each shuffler randomly permutes $k \in \Omega(\ln^2(n))$ random ciphertexts, then shuffling terminates in $\mathcal{O}(n/k \cdot \ln(n) + \beta)$ steps with a constant success probability.*

To underline the practicality of our distributed shuffling protocol, we implemented our solution and we provide benchmarks, which show that shuffling is not only asymptotically, but also practically efficient.

1.2 Applications

1.2.1 Single Secret Leader Elections

In the single secret leader election (SSLE) problem, introduced by Boneh et al. [4], we have a public bulletin board and n parties 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 [16] and threshold fully homomorphic encryption [5] respectively. Both of these solutions are theoretically interesting, but concretely too inefficient to be useful in a practical setting.

The third presented solution is based on a distributed shuffling protocol. In their protocol, each of the n participants publishes a commitment c_i only they can open on the bulletin board. Then, each participant's commitment is assigned a bucket of size $\sqrt{n}$, which is shuffled once. The protocol guarantees that each entry in the output vector could come from $\sqrt{n}$ possible locations in the input vector. The authors mention that stronger anonymity guarantees may be achieved using Håstad's square shuffle [17, 18], but leave the analysis of such an approach as an explicit open question. Håstad's square shuffle is an algorithm that shuffles a vector of length n using shuffles of size $k = \sqrt{n}$ in a benign setting. The algorithm itself does not provide any security guarantees in a setting, where an adversary may track some of the commitments or where the adversary can adaptively prevent some of the shuffles from happening. Even worse, it is straightforward to design an adaptive adversary with a relatively small budget of allowed corruptions that can prevent certain commitments from being shuffled at all.

Using our distributed shuffling protocol, which works for various choices of k beyond just $k = \sqrt{n}$, we obtain a new SSLE protocol that is secure against adaptive adversaries, where the elected leader is hidden not only among $\sqrt{n}$ other participants, but instead among close to all n of them.

Ethereum Block Proposer Elections

A variant of the SSLE problem has recently been considered in the context of the Ethereum blockchain, where we are not only interested in electing one, but rather a ordered list of γ leaders. Two real-world efficiency constraints are important to point out here. Every shufflers needs to speak in a timely manner, yet at the same time they need to provide zero-knowledge proofs attesting the correctness of their performed shuffle. These two constraints mean that no shuffler has enough time to permute the full vector at once.

The currently proposed protocol [14] for potential deployment in Ethereum is effectively a direct implementation of Håstad's square shuffle along with some other minor steps that are not relevant for the discussion here. The protocol is heuristically claimed to be secure against non-adaptive corruptions. However, the protocol is only discussed in an informal model and no security proofs are provided. Similarly to the plain square shuffle of Håstad, the proposed construction is not secure against an adaptive adversary that may adaptively target specific shufflers during the protocol execution. Especially in the context of a blockchain, where shufflers are known entities, and an adversary that may have the ability to target some of them adaptively, we believe that a stronger adaptive security notion and provable security guarantees are of crucial importance.

Our distributed shuffling protocol, which provides *provable* security guarantees against a stronger adversary, can be used as a direct replacement of the current proposal. Our experimental results show that the efficiency of our protocol is comparable to the current proposal of Ethereum. We discuss this application in more depth in Section 5.

1.3 Related Works

Multiple research domains are related to our work here.

1.3.1 Benign Shuffling

A series of existing works [27, 12, 1, 17, 18, 25, 22] 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 and it is unclear what security can be achieved in the presence of an adversarial entity.

Conceptually, the work of Diaconis and Shahshahani [12], which considers shuffling a deck of cards by repeatedly picking two random cards and switching them, is closest to our algorithm. In their work, the authors are interested in determining the required number of rounds until the resulting permutation looks close to uniformly random to a distinguisher that does not see which cards were swapped. In contrast to their work, we want to determine the required number of shuffles of size k until any uncorrupted card is at an unpredictable location, even if the adversary gets to see all subsets of k elements that were shuffled in the protocol.

1.3.2 Single Secret Leader Elections

After the first three initial approaches for solving the SSLE problem by Boneh et al. [4], an alternative solution based on functional encryption was proposed by Catalano, Fiore, and Giunta [9]. Their solution has many attractive properties, but requires an expensive initial setup to be performed between the parties participating in the elections. In situations where elections are performed periodically and many participants may join or leave between elections, the setup needs to be repeatedly renewed. Shuffling based solutions on the other hand, gracefully deal with joining and leaving participants, since no setup is required.

In a recent independent work by Catalano, Fiore, and Giunta [10], the authors also study SSLE in the presence of an adaptive adversary. Their work focuses on providing a full formalization of the SSLE problem in the universal composability framework [8]. The authors provide a solution based on shuffling that requires each shuffler to speak multiple times and to permute the full vector. In contrast to their work, we focus on distributed shuffling protocols, where shufflers have bounded memory and only speak once. We believe that our model is closer to how shufflers would actually operate in a real blockchain like Ethereum.

2 Preliminaries

2.1 Notation

We write $[n]$ to denote the set $\{1, \ldots, n\}$. We denote the computational security parameter by λ. 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 write $A^{\mathcal{O}(\cdot)}$ to denote algorithm A with oracle access to algorithm $\mathcal{O}$.

2.2 Encryption Schemes

We define the minimal security properties that are sufficient for proving our shuffling algorithm secure in our model. For the remainder of this work, we focus on shuffling a vector of ciphertexts, but all of our results easily carry over to commitments.

▶ **Definition 2.** *A public-key encryption scheme* $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a comprised of the following algorithms:*

$(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)$*: The key generation algorithm takes the security parameter* 1^λ *as input and outputs a public encryption key* ek *and a secret decryption key* dk*.*

$c \leftarrow \mathbf{Enc}(\mathsf{ek}, m)$*: The encryption algorithm takes the key* ek *and a message* m *as input and outputs a ciphertext* c*.*

$m \leftarrow \mathbf{Dec}(\mathsf{dk}, c)$*: The decryption algorithm takes key* dk *and ciphertext* c *as input and outputs message* m*.*

▶ **Definition 3** (Semantic Security)**.** *We say* $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is semantically secure, if for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{r}(\mathsf{ek},\mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)\\ (m_0,m_1) \leftarrow \mathcal{A}(\mathsf{ek})\\ b \leftarrow \{0,1\}\\ b^* \leftarrow \mathcal{A}(\mathsf{Enc}(\mathsf{ek},m_b))\end{array} : b = b^*\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the probability is taken over the uniform random coins of the adversary, the key generation, and the encryption algorithm.

The input to our distributed shuffling algorithm will be a vector of ciphertexts, where each one is encrypted under a different public key. To be able to meaningfully shuffle this vector, we require that ciphertexts under different keys are indistinguishable from each other. This notion of key privacy was first considered by Bellare et al. [3]. We use slightly weaker formalization of key privacy that is sufficient for our purposes.

▶ **Definition 4** (Key Privacy)**.** *We say a semantically secure encryption scheme* $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is key private, if for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{r}(\mathsf{ek}_0,\mathsf{dk}_0) \leftarrow \mathsf{Gen}(1^\lambda)\\ (\mathsf{ek}_1,\mathsf{dk}_1) \leftarrow \mathsf{Gen}(1^\lambda)\\ m \leftarrow \mathcal{A}(\mathsf{ek}_0,\mathsf{ek}_1)\\ b \leftarrow \{0,1\}\\ b^* \leftarrow \mathcal{A}(\mathsf{Enc}(\mathsf{ek}_b,m))\end{array} : b = b^*\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the probability is taken over the uniform random coins of the adversary, the key generation, and the encryption algorithms.

One possible instantiation of an encryption scheme with the desired properties is the ElGamal cryptosystem [13].

2.3 Local Shuffling Algorithms

The focus of our work lies in answering how to shuffle the elements of a vector of length n through the use of shuffle operations that permute k many elements at a time. To abstract away the concrete shuffling procedure that is used by any shuffler locally, we define an idealized function `Shuffle` that takes k ciphertexts as input and produces a fresh list of k ciphertexts that commit to the same multi-set of messages. In practice, the local shuffling procedure would be realized by combining a re-randomizable encryption or commitment scheme with an appropriate non-interactive zero-knowledge proof that attests the correctness of the performed shuffle. If the list were to contain ElGamal ciphertexts, then efficient shuffling arguments of Bayer and Groth [2] or Bünz et al. [7] could be used. If the list were to contain pedersen commitments [24], then efficient shuffling arguments of Bünz et al. [7] or Hoffmann et al. [19] could be used.

3 Model

In this section, we define the formal model within which we will present and analyze our distributed shuffling protocol. In our setting, we have a public bulletin board, where parties can post authenticated messages that are visible to all other parties. The messages are authenticated in the sense that each message on the bulletin board can be traced to its sender. In the beginning, the only thing written on the message board are ciphertxts $c_1, \ldots, c_n$, where $c_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, m_i)$ for some m_i for $i \in [n]$. The parties $P_1, \ldots, P_T$, also known as the shufflers, speak one after another by posting messages on the bulletin board. To compute their messages, each shuffler reads at most k ciphertexts, locally shuffles them using the `Shuffle` procedure, and writes the permuted vector of k ciphertexts (along with possibly auxiliary information) back on the bulletin board. At the end of the protocol execution, after all T shufflers have spoken, ciphertexts $\tilde{c}_1, \ldots, \tilde{c}_n$, which encrypt the same multiset as the input vector, should be written on the bulletin board. We call such a protocol Π a (T, n, k)-shuffle.

3.1 Corruptions

The shuffling protocol runs in the presence of adversarial behavior. The PPT adversary $\mathcal{A}$ 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, the adversary is corrupting α ciphertexts. For each corrupted ciphertext, the adversary learns the corresponding decryption key and can thus "track their positions" throughout the shuffling procedure. Additionally, the adversary is allowed to corrupt β shufflers in a fully adaptive manner, meaning that at the start of every round $i \in [T]$, the adversary is allowed to decide whether or not to corrupt shuffler P_i on the fly, as long as the total number of corrupted shufflers is at most β.

A corrupt shuffler can perform an arbitrary chosen, but valid permutation on an arbitrary choice of at most k ciphertexts. In principle, we do not need to assume that the adversary honestly permutes k ciphertexts or that she would even honestly report, which ciphertexts she touched. Both of these issues are easily resolved via standard non-interactive zero-knowledge arguments attesting the correctness of the shuffle. To avoid explicitly talking about such arguments, we simply restrict the adversary in her behaviour.

3.2 Definitions

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

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

$$\Pr\left[\begin{array}{r} (\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{Gen}(1^\lambda)\ \forall i \in [n] \\ c_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, i)\ \forall i \in [n] \\ \vec{c} := (c_1, \ldots, c_n) \\ (z, \pi) \leftarrow \langle \Pi(\vec{c}; r), \mathcal{A}(\vec{c}; \tilde{r}) \rangle \end{array} : \pi \neq \perp \right] = 1,$$

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

$\Pi(c_1, \ldots, c_n)$

for $t \in [T]$:

S_t picks random $\{i_1, \ldots, i_k\} \subset [n]$

S_t computes $(\tilde{c}_{i_1}, \ldots, \tilde{c}_{i_k}) \leftarrow \texttt{Shuffle}(c_{i_1}, \ldots, c_{i_k})$

S_t publishes $(\tilde{c}_{i_1}, \ldots, \tilde{c}_{i_k})$

Figure 1 Distributed shuffling protocol.

▶ **Definition 6** (Security). *Let $\mathcal{A}$ be a PPT adversary that corrupts at most β shufflers. We say that an (T, n, k)-shuffle Π is (ϵ, δ)-secure in the presence of an (α, β)-adversary $\mathcal{A}$, if for all $I \subset [n]$ with $|I| \leq \alpha$, it holds that with probability at least $1 - \delta$ we have*

$$\Pr\left[\begin{array}{c} (\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{Gen}(1^\lambda)\ \forall i \in [n] \\ c_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, i)\ \forall i \in [n] \\ \vec{c} := (c_1, \ldots, c_n) : \pi(i) = j \wedge i \notin I \\ \vec{dk} := \{\mathsf{dk}_i \mid i \in I\} \\ ((i,j), \pi) \leftarrow \langle \Pi(\vec{c}; r), \mathcal{A}(\vec{c}, \vec{dk}; \tilde{r}) \rangle \end{array}\right] \leq \epsilon,$$

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

In the definition above, there exists a naive attacking strategy. The adversary could just guess a random pair (i, j) of indices with $i, j \notin I$, which means that the best security we can hope for is $\epsilon = 1/(n - |I|)$. If on the other hand, we achieve $\epsilon \leq C/(n - |I|)$ for some constant C, then this translates into the intuitive guarantee that any element in the output vector comes from at least $(n - |I|)/C$ possible locations in the input vector.

4 Construction

Our distributed shuffling protocol is conceptually very simple. Each round, a shuffler picks a random subset of k ciphertexts and permutes those. The main technical challenge is to prove that after a not too large number of rounds, this procedure will shuffle the input vector sufficiently well.

We note that all shufflers in our protocol act independently and do not coordinate who will shuffle which entries in the vector. For this reason, even a powerful adaptive adversary cannot do anything better than corrupting an arbitrary subset of β shufflers. Thus, the question of how big the number of rounds T has to be set to tolerate an adversary that corrupts β shufflers, effectively reduces to the question of how well the input vector is shuffled in $T - \beta$ rounds in the presence of an adversary that can corrupt no shufflers at all.

The formal protocol description is given in Figure 1 and we prove the following theorem.

▶ **Theorem 7.** *Let $\mathcal{A}$ be a PPT adversary that corrupts at most β shufflers. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a semantically secure and key private encryption scheme. For any $0 < \delta < 1/3$, if $T \geq 20(n/k)\ln(n/\delta) + \beta$ and $k \geq 256\ln^2(n/\delta)(1 - \alpha/n)^{-2}$, then the protocol in Figure 1 is a (ϵ, δ)-secure (T, n, k)-shuffle in the presence of a (α, β)-adversary, where $\epsilon = 2/(n - \alpha) + \mathsf{negl}(\lambda)$.*

Proof. Let $I \subset [n]$ with $|I| = \alpha$ be an arbitrary, but fixed subset of indices belonging to ciphertexts that are corrupted by the adversary. Let $H := [n] \setminus I$ be the indices of uncorrupted ciphertexts. Let hybrid hybrid_0 be the security game as stated in Definition 6. We consider hybrid hybrid_1, which is identical to hybrid_0 with the exception that $(\mathsf{ek}_i, \mathsf{dk}_i) := (\mathsf{ek}_1, \mathsf{dk}_1)$ for all $i \in H$. Indistinguishability of hybrid_0 and hybrid_1 follows from the key privacy of the underlying encryption scheme. In hybrid_2 we set $c_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, 1)$ for all $i \in H$. Indistinguishability of hybrid_1 and hybrid_2 follows from the semantic security of the underlying encryption scheme. At this point, we observe that in each invocation of `Shuffle` by an honest shuffler the adversary learns *nothing* about how the honest ciphertexts were permuted. To see this, we note that each honest ciphertext returned by `Shuffle` is identically distributed, encrypted under the same key, encrypting the same message.

Next, we observe that an adaptive adversary can not do anything better than corrupting an arbitrary set of β shufflers. To see this, observe that each shuffler chooses its subset of k ciphertexts independently, thus the distribution of permutations between input and output vector that is produced by our protocol is independent of which shufflers are corrupted by $\mathcal{A}$. For the remainder of the proof we determine the number T_H of honest shuffles that need to be performed, such that every ciphertext's location is hidden sufficiently well. Our protocol can then be run for $T \geq T_H + \beta$ rounds to be secure against β corrupt shufflers.

We now view the ciphertexts as a set of n cups, denoted $c_1, \ldots, c_n$. Of these n cups, the last α are *idle* and the first $n - \alpha$ are *active*. The cups may contain a non-negative amount of water.

Let $k \geq 2$. A T_H step k-way *mixing* consists of repeatedly selecting k cups uniformly at random (without replacement). If B denotes the set of selected cups, we then gather all water in active cups $c_i \in B$. The collected water is then distributed evenly among the active cups. This process is repeated for T_H steps. We call one such step a *mixing step*.

We say that a T_H step k-way mixing is successful if, for any c_i among the active cups, if we had placed 1 unit of water in c_i and 0 in all remaining cups, then at the end of the mixing, no cup contains more than $2/(n - \alpha)$ water. That is, regardless of which active cup we choose put 1 unit of water in, at the end of shuffling, no cup contains more than a factor 2 more water than if we had distributed all water uniformly among active cups.

▶ **Lemma 8.** *For any $0 < \delta < 1/3$, if $T_H \geq 20(n/k)\ln(n/\delta)$ and $k \geq 256\ln^2(n/\delta)(1-\alpha/n)^{-2}$, then a T_H step k-way mixing with α idle cups is successful with probability at least $1 - \delta$.*

Observe first that if a T_H step k-way mixing is successful, then if we perform another mixing step, the mixing remains successful. This is because the maximum amount of water in a cup cannot increase in a mixing step. Hence we prove the lemma for $T_H = 20(n/k)\ln(n/\delta)$ and note that it also implies the result for larger T_H.

In our proof, we first show that if c_1 has 1 unit of water and the remaining have 0, then with probability at least $1 - \delta/n$, it holds that after T steps that there is no cup with more than $2/(n - \alpha)$ units of water. A union bound over all $n - \alpha$ active cups that may contain the initial 1 unit of water completes the proof.

So consider the setup where c_1 has 1 unit of water and the remaining have 0. We define two undesirable events, such that if none of these events occur, the mixing is successful. To define the first of these events, let B_t be the indices of the cups selected for mixing in the t'th step.

Consider an execution of a T_H step k-way mixing. A *back-tracking from cup c_i* is a sequence of indices $i_1, \ldots, i_r \in [k]$, possibly with repetitions, such that the following holds: Initialize $b = i$, $j = 0$ and $t = T_H$. Repeat until $t = 0$: If cup c_b was selected for mixing in

step t, increment j and set b to be the index of the i_j'th cup in B_t (for some arbitrary but fixed ordering on cups). Decrement t and repeat (regardless of whether cup c_b was selected for mixing in step t).

A back-tracking thus specifies a "path" that starts with c_i and as we go backwards through the T_H mixing steps, whenever the current cup c_b is selected for mixing, the path proceeds to trace the next cup in the list. When j reaches r in the back-tracking, it must be the case that the currently traced cup c_b is not selected in any further mixing steps while decrementing t. The first undesirable event says that there is a short back-tracking:

- Event E_1: There is a back-tracking $i_1, \ldots, i_r$ with $r \leq 4 \lg_k n$.

To define the second event, let w_i^t denote the amount of water in cup c_i after t steps of mixing. We have $w_1^0 = 1$ and $w_i^0 = 0$ for $i \neq 1$. Also, let $A_t \subseteq B_t$ denote the indices of the active cups among B_t. Finally, let $W_t = \sum_{i \in A_t} w_i^{t-1}/|A_t|$ denote average amount of water in the cups selected in step t. By definition, we have $w_i^t = W_t$ for every $i \in A_t$. With these definitions in place, the second undesirable events says that we in some step perform a mixing that results in much water on average, yet none of the involved cups had significantly more water than the average:

- Event E_2: There is a step t where $W_t \geq 2/(n-\alpha)$ but $\max_{i \in A_t} w_i^{t-1} \leq k^{1/4} W_t$.

Success when none of E_1 and E_2 occur. We first show that a T_H step mixing is successful when none of the events E_1 and E_2 occur. For this, consider an unsuccessful mixing where E_2 did not occur. We claim that this implies that E_1 occurred. We thus need to show that an unsuccessful mix together with the fact that E_2 does not occur implies a short back-tracking. For this, let c_{i^*} be a cup such that $w_{i^*}^T > 2/(n-\alpha)$. Such a cup exists since the mixing is unsuccessful. We will now back-track from that cup. So let $i = i^*$, $b = i$ and initialize $t = T_H$. Also, let ω^t denote the amount of water in the cup c_b traced in step t. We thus have $\omega^{T_H} = w_{i^*}^{T_H} > 2/(n-\alpha)$. We will guarantee that the values ω^t are non-decreasing when we decrement t from T_H towards 0. For $t = T_H$ down to 0, if cup c_b is selected for mixing in step t, we know that $W_t = \omega^t \geq \omega^{T_H} > 2/(n-\alpha)$. Since E_2 did not occur, it must be the case that $\max_{h \in A_t} w_h^{t-1} > k^{1/4} W_t$. Let h^* be the index into B_t of the h obtaining this maximum water in step $t-1$. We append h^* to the constructed list of indices $i_1, \ldots, i_r$ in the back-tracking as well as append the step t to the list of steps $t_1, \ldots, t_r$. We then update b to h, set ω^{t-1} to $w_h^{t-1} \geq k^{1/4} \omega^t$ and decrement t. If c_b was not selected for mixing, we simply decrement t.

Since ω increases by a factor at least $k^{1/4}$ each time the traced cup c_b is selected for mixing, it must be the case that $\omega^0 \geq 2k^{r/4}/(n-\alpha)$ if the produced back-tracking has length r. Since no cup ever contains more than 1 unit of water, this implies $2k^{r/4}/(n-\alpha) \leq 1 \Rightarrow r < 4 \lg_k n$. This implies that the event E_1 occurs.

Probability of success. In the following two paragraphs, we will show that $\Pr[E_1] \leq 2\delta^{10}/n$ and $\Pr[E_2] \leq \delta^2/n$. A union bound and the fact that $\delta < 1/3$ implies that a T_H step k-way mixing is successful with probability at least $1 - \delta/n$ when c_1 has 1 unit of water. As mentioned earier, a union bound over all $n - \alpha$ choices of the cup with 1 unit of water completes the proof. What remains is thus to bound the probability of E_1 and E_2.

There is a short back-tracking (Event E_1). To rule out the existence of a short back-tracking, consider a fixed value of $r \leq 4 \lg_k n$. For any such r, there are no more than $k^r \leq n^4$ choices for $i_1, \ldots, i_r$ and n choices for i. For any such choice, there are no more than $\binom{T_H}{r} \leq T_H^r$ choices for the steps $t_1, \ldots, t_r$ where j is decremented (the traced cup

is selected for mixing). Fix any such r, $i_1, \ldots, i_r$ and $t_1, \ldots, t_r$. For this to be a valid back-tracking, it must hold for all steps $t \notin \{t_1, \ldots, t_r\}$ that the cup c_b traced in that step is not selected for mixing. Since the mixing steps are independent, this happens with probability precisely $(1 - k/n)$ independently of the random choices in steps $t+1, \ldots, T_H$. For all steps $t \in \{t_1, \ldots, t_r\}$, it must be the case that the cup c_b traced in that step is selected for mixing. Again by independence, this happens with probability precisely k/n independently of the random choices in steps $t+1, \ldots, T_H$. For the fixed choice of i, r, $i_1, \ldots, i_r$ and $t_1, \ldots, t_r$, the probability that these form a valid back-tracking is thus no more than $(1-k/n)^{T_H - r}(k/n)^r \leq \exp(-(T_H - r)k/n)(k/n)^r$. We have $T_H = 20(n/k)\ln(n/\delta)$ and $k \leq n$, thus $r = 4\lg_k n \leq T_H/2$ and the probability is no more than $\exp(-T_H k/(2n))(k/n)^r = \exp(-10\ln(n/\delta))(k/n)^r = (k/n)^r(\delta/n)^{10}$. A union bound over all possible back-trackings of length $r \leq 4\lg_k n$ shows that

$$\begin{aligned}
\Pr[E_1] &\leq \sum_{r=0}^{4\lg_k n} n^5 T_H^r (k/n)^r (\delta/n)^{10} \\
&= \sum_{r=0}^{4\lg_k n} n^5 (20(n/k)\ln(n/\delta))^r (k/n)^r (\delta/n)^{10} \\
&= \sum_{r=0}^{4\lg_k n} n^5 (20\ln(n/\delta))^r (\delta/n)^{10}.
\end{aligned}$$

For $k \geq 40\ln(n/\delta)$, this is no more than

$$\begin{aligned}
\sum_{r=0}^{4\lg_k n} n^5 (20\ln(n/\delta))^r (\delta/n)^{10} &\leq \\
\sum_{r=0}^{4\lg_k n} n^5 (k/2)^r (\delta/n)^{10} &\leq \\
\sum_{r=0}^{4\lg_k n} 2^{-r} n^5 k^{4\lg_k n} (\delta/n)^{10} &= \\
2\delta^{10}/n.
\end{aligned}$$

A mix with much water, but no full cup (Event E_2). Let us first consider a fixed step t and condition on a fixed cardinality a of A_t and an arbitrary execution of the first $t-1$ steps. If we let $E'_{2,t}$ denote the event that $\max_{i \in A_t} w_i^{t-1} \leq k^{1/4} W_t$ and $E''_{2,t}$ the event that $W_t \geq 2/(n-\alpha)$. We now wish to bound $\Pr[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a]$. For this, we further split $E'_{2,t}$ and $E''_{2,t}$ into smaller events. Let $E'_{2,t,\xi}$ denote the event that $\max_{i \in A_t} w_i^{t-1} \leq 2k^{1/4}\xi$ and $E''_{2,t,\xi}$ the event $W_t \geq \xi$ and consider values of $\xi = 2^i/(n-\alpha)$ for $i = 1, \ldots, \lg_2 n$. We claim that

$$\Pr[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a] \leq \Pr\left[\cup_{i=1}^{\lg_2 n} (E'_{2,t,2^i/(n-\alpha)} \cap E''_{2,t,2^i/(n-\alpha)}) \mid |A_t| = a\right].$$

To see this, note that when $E''_{2,t}$ occurs, there is a maximal $1 \leq i < \lg_2 n$ for which $2^i/(n-\alpha) \leq W_t \leq 2^{i+1}/(n-\alpha)$. When $E'_{2,t}$ also occurs, this further implies $\max_{i \in A_t} w_i^{t-1} \leq 2^{i+1}k^{1/4}/(n-\alpha)$. That is, both of the events $E'_{2,t,2^i/(n-\alpha)}$ and $E''_{2,t,2^i/(n-\alpha)}$ occur. By a union bound, we thus have

$$\begin{aligned}
\Pr[E'_{2,t} \cap E''_{2,t}, |A_t| = a] &\leq \\
\sum_{i=1}^{\log_2 n} \Pr[E''_{2,t,2^i/(n-\alpha)} \cap E'_{2,t,2^i/(n-\alpha)} \mid |A_t| = a] &\leq \\
\sum_{i=1}^{\log_2 n} \Pr[E''_{2,t,2^i/(n-\alpha)} \mid E'_{2,t,2^i/(n-\alpha)}, |A_t| = a]
\end{aligned}$$

Next, we recall that each shuffler picks a uniformly random subset of cups to mix. If we condition this choice on $E'_{2,t,\xi}$ and $|A_t| = a$, then A_t is distributed as a uniform sample of a elements without replacement from the set of active cups c_i where $w_i^{t-1} \leq 2k^{1/4}\xi$. Furthermore, recall that we started with one cup containing one unit of water and all other cups being empty. If we were to sample a cup uniformly at random among all active cups, then the expected amount of water in a sampled cup would be precisely $1/(n-\alpha)$. Conditioning on $E'_{2,t,\xi}$ removes the most full cups and hence the expected amount of water in each sampled cup may only decrease when conditioning on $E'_{2,t,\xi}$. It follows from Hoeffding's inequality for sampling without replacement that for any $\xi \geq 2/(n-\alpha)$, we have

$$\begin{aligned}
\Pr[E''_{2,t,\xi} \mid E'_{2,t,\xi}, |A_t| = a] &= \\
\Pr[|W_t| \geq \xi \mid E'_{2,t,\xi}, |A_t| = a] &\leq \\
\Pr[|W_t - \mathbb{E}[W_t]| \geq \xi - 1/(n-\alpha) \mid E'_{2,t,\xi}, |A_t| = a] &\leq \\
\Pr[|W_t - \mathbb{E}[W_t]| \geq \xi/2 \mid E'_{2,t,\xi}, |A_t| = a] &\leq \\
2\exp\left(-\frac{2(a\xi 2/2)^2}{a(2k^{1/4}\xi)^2}\right) &= \\
2\exp\left(-a/(8\sqrt{k})\right).
\end{aligned}$$

Thus

$$\Pr[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a] \leq 2\lg_2(n)\exp(-a/(8\sqrt{k})).$$

Using this inequality, we then observe that

$$\begin{aligned}
&\Pr\left[E'_{2,t} \cap E''_{2,t}\right] \\
=&\sum_{a=0}^{k} \Pr\left[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a\right] \Pr\left[|A_t| = a\right] \\
=&\sum_{a=0}^{\frac{k(1-\alpha/n)-2}{2}} \Pr\left[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a\right] \Pr\left[|A_t| = a\right] \\
&+\sum_{a=\frac{k(1-\alpha/n)}{2}}^{k} \Pr\left[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a\right] \Pr\left[|A_t| = a\right] \\
\leq&\sum_{a=0}^{\frac{k(1-\alpha/n)-2}{2}} \Pr\left[|A_t| = a\right] \\
&+\sum_{a=\frac{k(1-\alpha/n)}{2}}^{k} 2\lg_2(n)\exp\left(-(k(1-\alpha/n)/2)/(8\sqrt{k})\right) \Pr\left[|A_t| = a\right] \\
\leq&\Pr\left[|A_t| \leq \frac{k(1-\alpha/n)}{2}\right] + 2\lg_2(n)\exp\left(-(k(1-\alpha/n)/2)/(8\sqrt{k})\right)
\end{aligned}$$

To conclude the proof, we would now like to argue that both of the terms in the last inequality above are small. We observe that B_t is a uniform sample without replacement from the n cups and thus we have that $\mathbb{E}[|A_t|] = k(1-\alpha/n)$. Using the Chernoff bound for sampling without replacement and assuming $k \geq 16\ln(n/\delta)(1-\alpha/n)^{-1}$, we get

$$\Pr[|A_t| \leq (1/2)k(1-\alpha/n)] \leq \exp(-k(1-\alpha/n)/8) \leq (\delta/n)^2.$$

Similarly, for $k \geq 256\ln^2(\delta/n)(1-\alpha/n)^{-2}$, we have that

$$\begin{aligned} &2\lg_2(n)\exp(-(1/2)k(1-\alpha/n)/(8\sqrt{k})) \\ =&2\lg_2(n)\exp(-\sqrt{k}(1-\alpha/n)/16) \\ \leq&2\lg_2(n)(\delta/n)^2 \end{aligned}$$

Thus for $k \geq 256\ln^2(\delta/n)(1-\alpha/n)^{-2}$, we have

$$\Pr[E'_{2,t} \cap E''_{2,t}] \leq 3\lg_2(n)(\delta/n)^2.$$

A union bound over all T_H then implies

$$\Pr[E_2] \leq 3T_H\lg_2(n)(\delta/n)^2.$$

There are $T_H = 20(n/k)\ln(n/\delta)$ choices for t and for $k \geq 256\ln^2(n/\delta)$, we have that

$$\Pr[E_2] \leq \delta^2/n,$$

which concludes the proof. ◀

5 Ethereum's Block Proposer Elections

One particular real-world application that can benefit from our shuffling protocol, is Ethereum's block proposer election. In the following, we provide a high-level idea of this election process and we refer the interested reader to the current proposal [14] for more details. In this setting, we have commitments[1] $(c_1, \ldots, c_n)$, where $n = 2^{14}$ and where c_i belongs to some identity i, who is the only entity that can open the commitment. These identities need to be arranged in a random secret order. Once this is done, the first γ owners of the commitments reveal themselves in order of the output list and perform some consensus related action that is not relevant for us. That is, the first identity in the output list is the first block proposer, the second identity the second proposer and so on. From a security perspective, one would like to ensure that an adversary that corrupts α identities, β of the shufflers, and gets to see some of the proposers that already revealed themselves, cannot guess the identity of the next honest block proposer.

In order to obtain the random secret ordering, in the current proposal, a sequence of shufflers are effectively executing Håstad's square shuffle [17, 18], i.e. there $k = \sqrt{n}$, interspersed with some additional public permutation steps. The current proposal is purely heuristic, is only described in an informal model, and does not have a security proof. It is not secure against an adversary that can corrupt shufflers adaptively.

[1] We note again that all of our results work equally well for vectors of commitments.

Our approach can be used to obtain a secret random ordering of the block proposers with stronger security guarantees and, in particular, with provable security guarantees. We note, however, that in our model we do not consider parts of the performed permutation to be revealed once the shuffling protocol is finished. Luckily our analysis can easily be amended to account for this.

In the proof of Theorem 7, we assumed that a fixed number of cups, denoted α, 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 α marked cups. In the second phase, we choose a uniform random subset of γ 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 corrupting α ciphertexts and then revealing γ random ciphertexts at the end. Let η be the number of idle cups.

Once the idle and active cups have been chosen, we run the water mixing process as in the proof of Theorem 7. We now bound the probability of seeing an active cup with more than $2/(n-\eta)$ units of water after T steps of mixing. We first bound the probability of seeing many idle cups. For this, notice that the first phase marks precisely α cups. For the second phase, the number of newly marked cups can be bounded by observing that the γ samples without replacement each picks a cup already marked in the first phase with probability precisely α/n (when looking at the marginal distribution of the cup). It follows by a Hoeffding bound for sampling without replacement that the number of newly marked cups in the second phase, denoted ζ, satisfies:

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

Setting $\ell = \sqrt{\gamma \ln(n/\delta)}$ bounds the above by δ^2/n^2. Thus with probability at least $1-\delta^2/n^2$, we have

$$\begin{aligned}\eta &\leq \alpha + \zeta \leq \alpha + \ell + (1-\alpha/n)\gamma \\ &= \alpha + \gamma - \alpha\gamma/n + \sqrt{\gamma \ln(n/\delta)}.\end{aligned}$$

A union bound together with Lemma 8 invoked with $\delta' = \delta/(2n)$ gives us that with probability at least $1 - \delta^2/n^2 - \delta/n$, there is no index i with $w_i^T \geq 2/(n-\eta)$. Note that the above analysis above is for a fixed number γ of revealed output locations. Doing a union bound over all $\gamma' \leq \gamma$ shows that the probability that throughout the revealing any of γ additional locations, that there is ever an input cup z whose output destination can be predicted with probability greater than $2/(n-\eta)$ is at most $n \cdot (\delta/(2n) + \delta^2/n^2) \leq \delta$.

6 Experiments

In this section, we perform numerical experiments to precisely determine the practical constants in our distributed shuffling protocol. We consider different sets of parameters. Since adversarially corrupt shufflers in our protocol are as bad as just no shuffle being performed, we simply measure the number of required honest shuffles, until the desired security guarantees are achieved. More precisely, if T_H honest shuffles are sufficient, then running our protocol for T rounds is secure against $\beta = T - T_H$ many corrupted shufflers.

In each experiment, we run the water mixing process from the proof of Lemma 8 with varying values for n, k, and α. For each fixed set of parameters the benchmark is repeated 100 times. In every round of an experimental run, we check whether any cup has too much water. If it does, then this run of the experiment for this round is considered to be failing. The

Table 1 Results of our numerical experiments for determining the number $T - \beta$ of honest shuffles that is needed for successfully shuffling with different sets of parameters.

#	n	k	α/n						
1	2^{14}	128	1/4	δ	0.8	0.6	0.4	0.2	0
				$T - \beta$	713	839	927	988	1804
2	2^{14}	256	1/4	δ	0.8	0.6	0.4	0.2	0
				$T - \beta$	337	398	452	502	627
3	2^{14}	512	1/4	δ	0.8	0.6	0.4	0.2	0
				$T - \beta$	199	229	254	278	438
4	2^{14}	128	1/2	δ	0.8	0.6	0.4	0.2	0
				$T - \beta$	874	955	1080	1204	1853

fraction of failing simulations in a given round, denoted by δ, is an unbiased estimate of the true probability that the adversary can determine the position of a uncorrupted ciphertext with probability greater than $2/(n - \alpha)$ in that round.

The result of our benchmarks are summarized in Figure 1. Even in a highly adversarial setting, where 1/2 of all elements in the vector are corrupted and the local shuffle size is as small as $k = 128$, our protocol successfully distributes the water after less than 2000 rounds for a vector of length $n = 2^{14}$. In the context of Ethereum's block proposer elections, we have $T = 2^{13}$ time slots for one election and thus one can tolerate a fraction of around 3/4 of corrupted shufflers.

References

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

2 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.

3 Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 566–582. Springer, 2001.

4 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.

5 Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *Annual International Cryptology Conference*, pages 565–596. Springer, 2018.

6 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.

7 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.

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

9 Dario Catalano, Dario Fiore, and Emanuele Giunta. Efficient and universally composable single secret leader election from pairings. Cryptology ePrint Archive, Paper 2021/344, 2021. URL: `https://eprint.iacr.org/2021/344`.

10 Dario Catalano, Dario Fiore, and Emanuele Giunta. Adaptively secure single secret leader election from ddh. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, 2022.

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

12 Persi Diaconis and Mehrdad Shahshahani. Generating a random permutation with random transpositions. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(2):159–179, 1981.

13 Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

14 Ethereum. Whisk: A practical shuffle-based ssle protocol for ethereum. *Accessed 09/09/2022*, 2022.

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

16 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 40–49, 2013.

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

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

19 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.

20 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.

21 Gregory Maxwell. Coinjoin: Bitcoin privacy for the real world. *Accessed 09/09/2022*, 2013.

22 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.

23 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.

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

25 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.

26 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.

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