On Publicly-Accountable Zero-Knowledge and Small Shuffle Arguments

Nils Fleischhacker[1⋆] and Mark Simkin[2⋆⋆]

[1] Ruhr University Bochum, Bochum, Germany
[2] Aarhus University, Aarhus, Denmark

Abstract. Constructing interactive zero-knowledge arguments from simple assumptions with small communication complexity and good computational efficiency is an important, but difficult problem. In this work, we study interactive arguments with noticeable soundness error in their full generality and for the specific purpose of constructing concretely efficient shuffle arguments.

To counterbalance the effects of a larger soundness error, we show how to transform such three-move arguments into publicly-accountable ones which allow the verifier to convince third parties of detected misbehavior by a cheating prover. This may be particularly interesting for applications where a malicious prover has to balance the profits it can make from cheating successfully and the losses it suffers from being caught.

We construct interactive, public-coin, zero-knowledge arguments with noticeable soundness error for proving that a target vector of commitments is a pseudorandom permutation of a source vector. Our arguments do not rely on any trusted setup and only require the existence of collision-resistant hash functions. The communication complexity of our arguments is *independent* of the length of the shuffled vector. For a soundness error of $2^{-5} = 1/32$, the communication cost is 153 bytes without and 992 bytes with public accountability, meaning that our arguments are shorter than shuffle arguments realized using Bulletproofs (IEEE S&P 2018) and even competitive in size with SNARKs, despite only relying on simple assumptions.

1 Introduction

Zero-knowledge arguments allow a prover to convince a verifier of the truth of a statement without leaking any additional information. Such arguments are a fundamental building block, ubiquitous in cryptography, with various applications in both theory and practice.

The quality of an argument system can be measured in several different ways. One of the most important quality measures is the size of the argument, i.e. how many bits the prover needs to exchange with the verifier to convince them of the

⋆ Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.
⋆⋆ Supported by a DFF Sapere Aude Grant 9064-00068B.

validity of the statement. Minimizing this measure is important for real-world applications, where the statements itself may be over several gigabytes large and where communicating large amounts of data over a wide area network can quickly turn into the main efficiency bottleneck. Another important measure is the computational efficiency of both prover and verifier. We would like our argument to incur as little computational overhead on both parties as possible. Finally, we would also like our arguments to rely on simple and well-studied assumptions. Arguments that rely on highly structured or even non-falsifiable assumptions may be prone to cryptanalysis, those that rely on more popular number-theoretic assumptions, like the discrete logarithm or the factoring assumption, can be broken by quantum computers, and arguments that require a common reference string need to rely on a trusted third party that has to generate this string.

One particularly popular class of zero-knowledge arguments are those that enable a prover to convince a verifier that two vectors of commitments or encryptions contain the same multiset of plaintext messages without revealing the messages themselves or the permutation between the two vectors. Shuffle arguments are used in applications like e-voting protocols [42], anonymous communication systems [18, 38], decentralized online poker [10], cryptocurrencies [20, 22], and others.

The idea of shuffle arguments originates in the work of Chaum on mix-nets [18] and the first constructions were presented by Sako and Kilian [45] and Abe [1, 2, 3]. These, as well as early subsequent works [25, 42, 30, 34], all had argument sizes, which were linear in the size ℓ of the permuted vector. The first sublinear shuffle argument, with an argument size of $\mathcal{O}\left(\ell^{2/3}\right)$, was presented by Groth and Ishai [33]. Following this work, Groth and Bayer [31, 9] presented arguments with an argument size of $\mathcal{O}\left(\sqrt{\ell}\right)$ and recently Bünz et al. [17] showed how to obtain arguments, based on the discrete logarithm assumption, of size $\mathcal{O}\left(\log \ell\right)$ via sorting circuits. A different line of works construct so called SNARKs [37, 40], which are constant-sized arguments for arbitrary statements. Unfortunately, SNARKs inherently rely on strong non-falsifiable assumptions [26], require a trusted setup, and are computationally expensive for the prover. Zero-knowledge arguments based on the MPC-in-the-head technique [35, 5] do not require any trusted setup, base their security solely on the existence of collision-resistant hash functions, but have a proof size of $\mathcal{O}\left(\sqrt{\ell}\right)$. Interactive proofs based on probabilistically checkable proofs [37] only rely on collision-resistant hash functions, have proofs of size $\mathcal{O}\left(\log \ell\right)$, but are prohibitively expensive from a computational perspective

Given this state of the art, it is evident that constructing small shuffle arguments, and more generally arbitrary arguments, from simple assumptions with good computational efficiency is a challenging, but important task. In this work, we ask:

Can we construct shuffle arguments of size $o(\log \ell)$ with good computational efficiency from simple assumptions that satisfy a relaxed, but still meaningful security notion?

Answering this question is of practical importance. In certain real-world scenarios, arguments that satisfy the strongest possible security notion can simply be too inefficient. In these cases more efficient arguments that satisfy a weaker security notion may provide an important trade-off between efficiency and security.

1.1 Our Contribution

In this work, we study interactive arguments with noticeable soundness errors, i.e. arguments that allow a cheating prover to convince a verifier of a false statement with some noticeable probability. Such arguments can still be useful in scenarios, where a malicious prover has to balance the profit that it can make from successfully cheating and the loss it has, when cheating is detected. Consider for instance a decentralized online poker game, where a malicious prover wins \$1 for every incorrect shuffle argument that is accepted by the verifier, but loses a \$100 security deposit if cheating is detected. In such a scenario, a soundness error as large as $1/2$ may be acceptable, since even then cheating is not profitable for a rationally behaving prover.

We study arguments with noticeable soundness error both in their full generality and for specific purpose of constructing concretely efficient shuffle arguments for pseudorandom shuffles. Concretely, we make the following contributions:

Publicly-Accountable Zero-Knowledge Arguments. To realize the idea of punishing cheating provers, we need to take care of two things. First, we need to ensure that a verifier, upon detecting a cheating attempt, obtains a publicly verifiable certificate that can be used to convince a third party auditor of the prover's malicious behavior. Secondly, we need to ensure that an honest prover cannot be falsely accused. We introduce the notion of *publicly-accountable zero-knowledge arguments* that formally model the two requirements above.

In the full version of this work, we show how to transform *any* three-move, honest-verifier zero-knowledge argument with a soundness error that is at least inversely polynomial in the security parameter into a publicly-accountable argument with only slightly larger communication complexity. This is achieved via two steps. We prove that those honest-verifier zero-knowledge arguments already satisfy full zero-knowledge and then show how such zero-knowledge arguments can be transformed into their publicly-accountable counterparts with the help of symmetric private information retrieval. In this version of the work we show how to make the shuffle arguments described below publicly accountable in a concretely efficient manner.

It is interesting to note that in contrast to sequential repetition, which is commonly used to make the soundness error negligible at the cost of a multiplicative factor that is linear in the security parameter, our transformation only incurs a small additive factor in terms of round and bandwidth complexity.

Shuffle Arguments for Pseudorandom Shuffles. Next, we focus on constructing efficient three-move (public-coin) honest-verifier zero-knowledge shuffle arguments with inversely polynomial soundness error. For this purpose, we make one additional observation that allows us to further simplify the problem we aim to solve. When looking at the majority of applications, where shuffle arguments are actually used, the concrete permutation between the input and the output vector is chosen at random. Most often, the goal is to simply hide the relation between entries in the input and the output vector, but the concrete permutation itself is irrelevant as long as it is sufficiently random. In e-voting, for example, users send their encrypted votes to an untrusted shuffling authority, which shuffles them to hide the voting preference of any specific user. In anonymous communication systems, users send messages through one or more shuffling authorities to some recipients and shuffling of the ciphertexts ensures that no outside observer can see which sender communicates with which recipient. Thus, we focus on shuffle arguments for pseudorandom shuffles instead of arbitrary shuffles.

We introduce the notion of *zero-knowledge arguments for partially fixed statements* and present conceptually simple interactive shuffling arguments satisfying this notion. The main idea behind our new notion is to consider statements that are only partially fixed, i.e. that consist of a fixed and a non-fixed part. The fixed part is known to both prover and verifier, whereas the non-fixed part is chosen by the prover. At the end of an interaction between prover and verifier, the verifier learns the full statement and is convinced of its correctness. For the specific case of shuffling, the fixed part is the initial vector of commitments and the non-fixed part is a permutation thereof. Our notion aims to capture the fact that we only care about the initial vector being permuted, but not about the concrete permutation that is used. Rather than requiring that zero-knowledge holds for all statements, we require that zero-knowledge holds for all partially fixed statements with a uniformly random non-fixed part. Our notion is, in spirit, similar to distributional zero-knowledge [27, 19], but focuses on a particular distribution over the statements.

For this notion, we present the first computationally efficient shuffle arguments for pseudorandom permutations, whose argument size is *independent* of the length of the vector that is being permuted. More specifically, we present public-coin, three-move arguments in the standalone model based on simple assumptions, such as collision-resistant hash functions.[3] The soundness error in our constructions can be set arbitrarily small as long as it remains inversely polyno-

[3] Naturally, our arguments are only useful for vectors of rerandomizable commitments/encryptions, which may require specific number-theoretic assumptions. The arguments itself, however, only rely on simple assumptions.

mial in the security parameter. The computational overhead of our construction grows with smaller soundness errors. We show how an arbitrary number of shuffle arguments can be batched without any additional communication cost.

We evaluate the practical efficiency of our constructions by providing concrete argument sizes when instantiated in the standard model. Our evaluation shows that, for a soundness error of 2^{-5}, the instantiation of our shuffle argument has a communication cost of 153 bytes without and 992 bytes with public accountability,

This is on the same order of magnitude as SNARKs such as [32] at 144 bytes and smaller than Bulletproofs even when the permuted vector of commitments[4] is reasonably short. The size of our argument is independent of the specific number of commitments being shuffled. The computational cost of shuffling an ℓ-length vector with soundness error $1/t$ is dominated by computing $t \cdot \ell$ rerandomizations of the shuffled commitments for both prover and verifier. In practice for Pedersen and similar commitments, the cost for the verifier can be reduced to roughly 2ℓ rerandomization at the cost of roughly doubling the communication complexity. A detailed description of this modification can be found in the full version of this paper.

Since the non-fixed part of the partially fixed shuffle statement is chosen randomly in each execution, it follows that the fully fixed statement will be different in each execution with high probability. For this reason, we cannot reduce the soundness error via sequential repetition. Due to the non-negligible soundness error, we can also not use the Fiat-Shamir transform [23] for making them non-interactive.

1.2 Comparison to SNARKs

In terms of size, our shuffle arguments are similar to SNARKs. Our underlying assumptions, however, are significantly weaker. We do not rely on non-falsifiable assumptions or a trusted setup. In exchange, we have a noticeable soundness error. As such, for the specific case of shuffle arguments, our work shows that the need for a trusted setup and non-falsifiable assumptions can be overcome in a practically efficient manner in applications that can tolerate a small, but noticeable soundness error.

1.3 Relation to Secure Computation with Covert Security

Our concept of publicly-accountable zero-knowledge arguments is strongly related to general two- and multiparty computation protocols with covert security and public verifiability [7, 6]. A protocol is said to be secure against covert adversaries and publicly verifiable, if an actively misbehaving party in the protocol execution is caught with some constant probability and the honest parties are

[4] For the sake of concreteness, we focus on commitments in this work. However, our arguments are also applicable to other primitives such as rerandomizable encryption schemes.

guaranteed to obtain a certificate that can be shown to third parties to incriminate the misbehaving party. Applying a generic secure computation compiler, like the one of Damgård, Orlandi, and Simkin [21], to transform zero-knowledge arguments into publicly-accountable ones can potentially work, but would result in arguments whose size has an exponentially worse dependence on the soundness error.

1.4 Technical Overview

In the following, we present the main ideas behind our public-coin, zero-knowledge shuffle argument and outline how it can be made publicly accountable. In the full version, we show how arbitrary public-coin arguments with a polynomially large challenge space can be made publicly accountable.

The Shuffle Argument. Initially, both prover and verifier are given an initial vector of commitments V. The goal of the prover is to choose some vector V' and convince the verifier that there exists some permutation π, such that $V' = \pi(V)$. To be precise, the permutation π here does two things. It first rerandomizes and then permutes all commitments in V. The high-level idea behind our construction is to let the prover choose a permutation π, which can be represented as a sequence of t pseudorandom permutations $\pi_1, \ldots, \pi_t$ in a space-efficient manner. The prover first computes V' by sequentially applying each permutation π_i for $1 \leq i \leq t$ and then sends a hash of the intermediate vectors V_i and V' to the verifier, who picks $t-1$ permutations that shall be opened. The prover sends descriptions of these $t-1$ permutations to the verifier. Skipping over some details, the verifier now uses V to check every permutation π_j with $j < i$ and V' to check every permutation π_j with $j > i$ by recomputing the intermediate vectors. Since π_i remains hidden from the verifier, it cannot learn the overall permutation π. A malicious prover can only convince the verifier of a false statement if it chose all π_j with $i \neq j$ as correct and π_i as an incorrect permutation. Thus the probability of a prover cheating successfully is $1/t$. An interesting open question is whether our approach can be modified to get a better dependence between t and the resulting soundness error.

If done naively, then the size of the argument described above is $\mathcal{O}(t \cdot \log(\ell!))$, since the prover has to send each permutation π_j for $1 \leq j \leq t$ separately. This can easily be brought down to $\mathcal{O}(t)$ by sending only short random seeds which can be expanded into a pseudorandom permutation using a regular pseudorandom generator. To further reduce the size of our argument, we use a puncturable pseudorandom functions (PPRF), which behaves like a regular PRF but has an additional algorithm $\mathsf{Puncture}$ that allows the holder of a secret key k to compute a key $k\{x\}$, which can be used to evaluate the PPRF on every point $x' \neq x$. Importantly, for a PPRF it holds that the key $k\{x\}$ does not reveal anything about its evaluation at the point x. Assuming that we have a PPRF $\mathcal{F}$ whose range is the set of all possible permutations, we can now succinctly represent π_j as the evaluation $\mathcal{F}(k, j)$ for $1 \leq j \leq t$. When the verifier asks to open all but

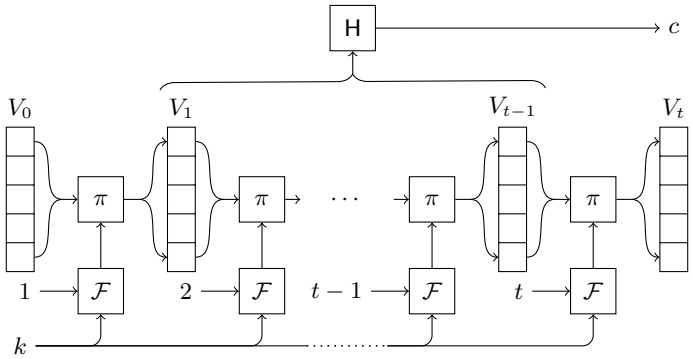

Fig. 1: The prover chooses the permutation and rerandomization by sampling a key k for a puncturable PRF. It then derives individual permutations and rerandomization factors for each stage by feeding the stage index through the PRF. The final result of performing the individual permutations forms the final shuffling and thus the prover-chosen part of the statement. The prover then hashes all of the intermediate permutations. The puncturable key k will allow the prover to partially open the computation of the intermediate permutations.

the i-th permutation, we return the punctured key $k\{i\}$ to the verifier. Using this approach, the size of the argument now mainly depends on the size of the punctured key and not directly on t. Using a PPRF based on the GGM construction [28], this brings down the size of the argument to $\mathcal{O}(\log t)$. Using a recent construction of PPRFs due to Aviram, Gellert, and Jager [8] we can make the size of our arguments even completely independent of t in the random oracle model. However, due to the large constants in Aviram et al.'s construction, this approach is only of theoretical interest. A visual illustration of our construction can be found in Figure 1.

Making the Argument Publicly Accountable. To make the shuffle argument publicly-accountable, we need to ensure that a cheating prover produces some form of self-incriminating evidence whenever it fails to cheat. Since we want this evidence to be publicly verifiable, we need to assume that there exists a public signature key pk that is associated with the prover, who holds the corresponding signing key sk.

Ideally, we would like the prover to sign the transcript of all exchanged messages at the end of each execution and send this signature to the verifier. If the prover were to always do this, then we would be done, since the verifier would obtain a signature incriminating the prover, whenever it attempts to cheat, but is caught; assuming the prover always sends some last message, even if it can not respond correctly. Obviously this does not work, since the prover can simply abort the execution without signing anything, when it receives a challenge that it does not like. Our idea is to let the verifier receive the prover's last message

corresponding to the verifier's challenge, without revealing the challenge to the prover. On a high-level, we achieve this through the use of symmetric private information retrieval, which enables a receiver holding an index $i \in \{1, \ldots, t\}$ to obtain a value x_i from the sender's input vector $X = (x_1, \ldots, x_t)$ in a manner that does not reveal i to the sender and does not reveal any x_j for $j \neq i$ to the receiver.

For the specific case of our shuffle arguments, the senders inputs will be a sequence of punctured keys and the receiver will retrieve one of them. We observe that in this instance the symmetric PIR can in fact be replaced by a very efficient oblivious key puncturing protocol implicit in the work of Boyle et al. [14].

2 Preliminaries

We denote by $\lambda \in \mathbb{N}$ the security parameter that is implicitly given as input to all algorithms in unary representation 1^λ. We denote by $\{0,1\}^\ell$ the set of all bit-strings of length ℓ. For a finite set S, we denote the action of sampling x uniformly at random from S by $x \leftarrow S$, and we denote the cardinality of S by $|S|$. An algorithm is efficient or PPT if it runs in time polynomial in the security parameter. If $\mathcal{A}$ is randomized then by $y := \mathcal{A}(x; r)$ we denote that $\mathcal{A}$ is run on input x and with random coins r and produces output y. If no randomness is specified, then it is assumed that $\mathcal{A}$ is run with freshly sampled uniform random coins. We write this as $y \leftarrow \mathcal{A}(x)$. A function $\mathsf{negl}(\lambda)$ is negligible if for every positive polynomial $\mathsf{poly}(\lambda)$ there exists an $N \in \mathbb{N}$ such that for all $\lambda > N$, $\mathsf{negl}(\lambda) \leq \frac{1}{\mathsf{poly}(\lambda)}$.

For two interactive Turing machines A and B we denote by $\langle A(a), B(b)\rangle$ the execution of the protocol between A and B an inputs a and b. We denote by $(t, s) \leftarrow \langle A(a), B(b)\rangle$ the outputs of A and B after the protocol execution respectively. In protocols where A does not receive an output, we write $s \leftarrow \langle A(a), B(b)\rangle$ to denote the output of B. We further denote by $\mathcal{T} := \langle A(a), B(b)\rangle$ the transcript resulting from the interaction.

2.1 Puncturable Pseudorandom Functions

Puncturable pseudorandom functions (PPRFs) can be constructed from one-way functions, where the key-length is $\mathcal{O}(\log |D|)$ and D is the input domain of the PRF [12, 36, 15]. Subsequent works have shown how to construct PPRFs with short keys from the strong RSA [8] and lattice-based assumptions [16].

Definition 1 (Puncturable PRFs). *The tuple* $(\mathcal{F}, \mathsf{Puncture})$ *of* PPT *algorithms is a secure puncturable pseudorandom function with key length* $\kappa(\lambda)$*, input length* $i(\lambda)$*, and range* $O(\lambda)$ *if the following conditions hold:*

Functionality: *For every* $\lambda \in \mathbb{N}$*,* $k \in \{0,1\}^{\kappa(\lambda)}$*,* $x, x' \in \{0,1\}^{i(\lambda)}$ *with* $x \neq x'$*, and* $k' \leftarrow \mathsf{Puncture}(k, x)$ *it holds that* $\mathcal{F}(k, x') = \mathcal{F}(k', x')$*.*

Pseudorandomness: *For any* PPT *adversary* $\mathcal{A}$ *it holds that*

$$\left| \begin{array}{l} \Pr[x \leftarrow \mathcal{A}(1^\lambda) : \mathcal{A}(\mathsf{Puncture}(k,x), \mathcal{F}(k,x)) = 1] \\ -\Pr[x \leftarrow \mathcal{A}(1^\lambda); y \leftarrow O(\lambda) : \mathcal{A}(\mathsf{Puncture}(k,x), y) = 1] \end{array} \right| \leq \mathsf{negl}(\lambda).$$

For our shuffle arguments (for vectors of length ℓ) we require a PPRF with range $\mathsf{Perm}_\ell \times \mathcal{R}^\ell$ where $\mathcal{R}$ is the randomness space of a perfectly and inversely rerandomizable commitment scheme and Perm_ℓ is the set of all $\ell!$ permutations over $\{0, \ldots, \ell-1\}$. To obtain a PPRF over this range, one can simply use a PPRF that outputs bit strings, potentially stretching the output using a pseudorandom generator, and combine it with a shuffling algorithm like the Fisher-Yates shuffle [24] by using the stretched output of the PPRF as the random tape of the shuffling algorithm.

2.2 Oblivious Key Puncturing

We formalize the notion of an oblivious key puncturing protocol (OPP) between a receiver R, who has a secret index i, and a sender S, who has a secret PRF key k. At the end of the protocol execution, the receiver should learn the key punctured at i, while the sender should learn nothing. An oblivious key puncturing protocol is effectively a special case of a symmetric PIR, where the sender's inputs are all possible punctured keys.

Definition 2 (Oblivious Key Puncturing). *Let* $(\mathcal{F}, \mathsf{Puncture})$ *be a secure puncturable PRF with key length* $\kappa(\lambda)$*, input length* $i(\lambda)$*, and range* $O(\lambda)$*. A pair of* PPT *algorithms* (S, R) *along with a setup algorithm* Setup *that outputs a* crs *is a secure receiver-extractable, oblivious key puncturing protocol for* $(\mathcal{F}, \mathsf{Puncture})$*, if the following conditions hold:*

Completeness: *For any* $k \in \{0,1\}^{\kappa(\lambda)}$ *and* $i \in \{0,1\}^{i(\lambda)}$*, it holds that*

$$\Pr\left[\begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda); k' \leftarrow \mathsf{Puncture}(k,i); \\ k'' \leftarrow \langle \mathsf{S}(\mathsf{crs}, k), \mathsf{R}(\mathsf{crs}, i) \rangle \end{array} : k' = k''\right] = 1.$$

Receiver Privacy: *For any* $i, i' \in \{0,1\}^{i(\lambda)}$ *and any malicious* PPT *sender* S^**, it holds that*

$$\left| \begin{array}{l} \Pr[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda); b \leftarrow \langle \mathsf{S}^*(\mathsf{crs}), \mathsf{R}(\mathsf{crs}, i) \rangle : b = 1] \\ -\Pr[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda); b \leftarrow \langle \mathsf{S}^*(\mathsf{crs}), \mathsf{R}(\mathsf{crs}, i') \rangle : b = 1] \end{array} \right| \leq \mathsf{negl}(\lambda),$$

where the probabilities are taken over the random coins of Setup*,* S^* *and* R*.*

Sender Simulation: *There exists a* PPT *simulator* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *such such that for any key* $k \in \{0,1\}^{\kappa(\lambda)}$ *and any malicious* PPT *receiver* R^* *it holds that*

$$\left| \begin{array}{l} \Pr[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda); b \leftarrow \langle \mathsf{S}(\mathsf{crs}, k), \mathsf{R}^*(\mathsf{crs}, 1^\lambda) \rangle : b = 1] \\ -\Pr\left[\begin{array}{r} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda); \\ b \leftarrow \langle \mathsf{Sim}_1^{\mathsf{Puncture}(k,\cdot)}(\mathsf{crs}, \mathsf{td}), \mathsf{R}^*(\mathsf{crs}, 1^\lambda) \rangle \end{array} : b = 1\right] \end{array} \right| \leq \mathsf{negl}(\lambda),$$

where Sim_1 *can query its oracle at most once and the probability is taken over the random coins of the involved parties.*

Remark 1. Protocols that need a (non-empty) CRS, require a trusted setup. For those protocols, using standard techniques, the trusted setup can be avoided at the cost of a constant number of additional rounds of communication.

It turns out that for the PPRF based on one-way functions [12, 36, 15], highly efficient instantiations of such an oblivious key puncturing protocol already exist implicitly in [14][5]. For a PPRF with domain D, the communication and computational complexity of their protocol is effectively that of $\log|D|$ invocations of an actively secure 1-out-of-2 oblivious transfer.

2.3 Commitments

Shuffle proofs are generally only of interest for rerandomizable commitment schemes. Our construction of shuffle proofs requires more than just perfect rerandomizability. Specifically we require that rerandomization can also be performed in reverse.

Definition 3 (Perfectly and Inversely Rerandomizable Commitments). *Let* $\mathcal{C} = (\mathsf{Setup}, \mathsf{Com})$ *be a commitment scheme with message space* $\mathcal{M}$ *and randomness space* $\mathcal{R}$. $\mathcal{C}$ *is perfectly and inversely rerandomizable, if there exist* PPT *algorithms* $\mathsf{Rerand}, \mathsf{Rerand}^{-1}$ *such that the following conditions hold:*

Perfect Rerandomization: *For every* $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$, $m \in \mathcal{M}$, *and* $c \leftarrow \mathsf{Com}(\mathsf{ck}, m)$*it holds that for a uniformly chosen* $r \leftarrow \mathcal{R}$, $\mathsf{Rerand}(\mathsf{ck}, c, r)$ *and* $\mathsf{Com}(\mathsf{ck}, c; r)$ *are distributed identically.*

Inverse Rerandomization: *For every* $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$, $m \in \mathcal{M}$, $r \in \mathcal{R}$, *and* $c \leftarrow \mathsf{Com}(\mathsf{ck}, m)$ *it holds that* $\mathsf{Rerand}^{-1}(\mathsf{ck}, \mathsf{Rerand}(\mathsf{ck}, c, r), r) = c$.

One example of a popular commitment scheme satisfying the properties described above is the Pedersen commitment scheme [43]. Note that since we assume perfect rerandomizability, it is guaranteed that for any ck, c, r_1, and r_2, there exists an r_3 such that $\mathsf{Rerand}(\mathsf{ck}, \mathsf{Rerand}(\mathsf{ck}, c, r_1), r_2) = \mathsf{Rerand}(\mathsf{ck}, c, r_3)$.

3 Zero-Knowledge Argument for Partially Fixed Statements

In a regular proof or argument system, the full statement is fixed a priori and given to both the prover and verifier which then run an interactive protocol between them. In contrast in an arguments for partially fixed statements only

[5] The authors prove (in Theorem 7 in [14]) that their construction satisfies a weaker notion than the one we defined here, but it can be easily seen that their construction satisfies our notion as well, when instantiated with an actively secure oblivious transfer.

a part x of the statement is fixed and the *prover gets to sample the rest of the statement* y together with auxiliary information aux that will allow the prover to efficiently prove that $(x, y) \in \mathcal{L}$. Note, that aux is not necessarily just a regular witness for (x, y). In fact, in our shuffle proof, a regular witness for the statement would merely be the permutation and rerandomization factors. However, the auxiliary information used by our prover is a highly compact representation of a decomposition of both the permutation and rerandomization. This also implies that a prover in such a system is not necessarily capable of proving all $(x, y) \in \mathcal{L}$, but merely a, potentially small, subset. However, the definition of zero-knowledge will imply that the full statement (x, y) chosen by the prover is indistinguishable from a uniform choice of $(x, y) \in \mathcal{L}$ conditioned on x. To formally define such argument systems, we first define partially fixable languages, as those languages where y can be efficiently uniformly sampled conditioned on x.

Definition 4 (Partially Fixable Languages). *Let X, Y be sets. Let $\mathcal{L} \subseteq X \times Y$ be an NP language consisting of pairs $(x, y) \in X \times Y$ with the corresponding NP-relation $\mathcal{R}$. For any $x \in X$ we denote by $\mathcal{L}_x$ the language $\mathcal{L}_x = \{(x, y') \mid y' \in Y \wedge (x, y') \in \mathcal{L}\}$. $\mathcal{L}$ is called partially fixable if for all $x \in X$ such that $\mathcal{L}_x \neq \emptyset$ it is possible to uniformly sample from $\mathcal{L}_x$ in expected polynomial time.*

We can now define argument systems for such languages.

Definition 5 (Arguments for Partially Fixed Statements). *Let $\mathcal{L} \subseteq X \times Y$ be a partially fixable language with the corresponding NP-relation $\mathcal{R}$. A probabilistic polynomial time two-stage prover $\mathsf{P} = (\mathsf{P}_0, \mathsf{P}_1)$ and a probabilistic polynomial time verifier V are said to be an interactive argument for partially fixed statements of $\mathcal{L}$ with soundness error ϵ if the following conditions hold:*

Completeness: *For any $x \in X$ with $\mathcal{L}_x \neq \emptyset$ it holds that*

$$\Pr[(y, \mathsf{aux}) \leftarrow \mathsf{P}_0(x); b \leftarrow \langle \mathsf{P}_1(x, y, \mathsf{aux}), \mathsf{V}(x, y) \rangle : (x, y) \in \mathcal{L} \wedge b = 1] = 1.$$

Soundness: *For any malicious probabilistic polynomial time prover P^* and any $(x, y) \notin \mathcal{L}$ it holds that*

$$\Pr[b \leftarrow \langle \mathsf{P}^*(1^\lambda), \mathsf{V}(x, y) \rangle : b = 1] \leq \epsilon + \mathsf{negl}(\lambda).$$

We are only interested in arguments that are zero knowledge. We define two flavors of zero-knowledge.

Definition 6 (Zero-Knowledge Arguments for Partially Fixed Statements). *Let (P, V) be an interactive argument for partially fixed statements of $\mathcal{L}$. The argument is said to be zero knowledge if there exists an expected polynomial time simulator Sim, such that for any (potentially malicious) polynomial time verifier V^*, all probabilistic polynomial time distinguishers $\mathcal{D}$, and all $x \in X$ with $\mathcal{L}_x \neq \emptyset$ it holds that*

$$\left| \begin{array}{l} \Pr[(y, \mathsf{aux}) \leftarrow \mathsf{P}_0(x); s \leftarrow \langle \mathsf{P}_1(x, y, \mathsf{aux}), \mathsf{V}^*(x, y) \rangle : \mathcal{D}(x, y, s) = 1] \\ -\Pr[(x, y) \leftarrow \mathcal{L}_x; s \leftarrow \mathsf{Sim}^{\mathsf{V}^*(x,y)}(x, y) : \mathcal{D}(x, y, s) = 1] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

where Sim has the power of rewinding V^*.

Definition 7 (Honest Verifier Zero-Knowledge). *Let* (P, V) *be an interactive argument for partially fixed statements of* $\mathcal{L}$*. The argument is said to be* honest verifier *zero knowledge if there exists an expected polynomial time simulator* Sim*, such that for all probabilistic polynomial time distinguishers* $\mathcal{D}$*, and all* $x \in X$ *with* $\mathcal{L}_x \neq \emptyset$ *it holds that*

$$\left| \begin{array}{r} \Pr[(y, \mathcal{T}) \leftarrow \textsc{Simu}(x) : \mathcal{D}(x, y, \mathcal{T}) = 1] \\ -\Pr[(y, \mathcal{T}) \leftarrow \textsc{Real}(x) : \mathcal{D}(x, y, \mathcal{T}) = 1] \end{array} \right| \leq \mathsf{negl}(\lambda),$$

where Real *and* Simu *are defined as follows*

$\textsc{Real}(x)$	$\textsc{Simu}(x)$
$(y, \mathsf{aux}) \leftarrow \mathsf{P}_0(x)$	$(x, y) \leftarrow \mathcal{L}_x$
$\mathcal{T} := \langle \mathsf{P}_1(x, y, \mathsf{aux}), \mathsf{V}(x, y) \rangle$	$\mathcal{T} \leftarrow \mathsf{Sim}(x, y)$
return $(y, \mathcal{T})$	**return** $(y, \mathcal{T})$

We note several important differences compared to regular argument systems. When defining an argument systems where the prover can choose part of the statement completeness can no longer be defined by simply quantifying over all valid statements. Instead, completeness explicitly specifies that the honest prover will always choose valid statements. Further, in the definition of zero-knowledge, it is not necessarily clear how y should be chosen in the simulated case. The definition above requires that y is chosen uniformly at random from $\mathcal{L}_x$ in this case as opposed to also being chosen by the prover. This has an important implication. Namely it implicitly requires the honest prover to choose y in a way that is computationally indistinguishable from uniform, since otherwise there exists a trivial distinguisher. Lastly we note that these definitions coincide with the standard zero-knowledge argument definitions, when $|\mathcal{L}_x| = 1$.

4 On Three-Move Public-Coin HVZK Arguments and Zero-Knowledge

In the following, we show that any three-move public-coin[6] argument with a polynomially large challenge space that satisfies (computational) HVZK is also (computationally) zero-knowledge against malicious verifiers. A corollary of this result is that our shuffle argument from Section 5, which we will prove to be HVZK, is automatically fully zero-knowledge.

Theorem 1. *Let* (P, V) *be some three-move public-coin* honest verifier *zero-knowledge argument for language* $\mathcal{L} \subseteq X \times Y$ *and let* $\mathcal{C}$ *be the associated challenge space. If* $|\mathcal{C}| \leq \mathsf{poly}(\lambda)$ *then* (P, V) *is also zero-knowledge against malicious verifiers.*

[6] We call a three-move argument system public-coin if the second message is a uniformly random bit-string.

Proof. Let V^* be an arbitrary malicious polynomial time verifier. Let Sim' be the *honest verifier* zero-knowledge simulator for the 3-move public-coin argument as specified in Definition 7. To prove the theorem, we specify a zero-knowledge simulator Sim that takes as input a statement (x, y), has blackbox access to V^*, and produces an output that is computationally indistinguishable from the output of V^* in a real protocol execution.

At first sight, the proof of the theorem statement may seem trivial. Intuitively, Sim picks a random challenge d, runs the simulator Sim' to obtain a transcript (e, d, z), feeds the first message e to V^* and if the verifier outputs a challenge d^* with $d^* = d$, then we are done and otherwise we simply restart this whole process until we guess the verifier's challenge correctly. Unfortunately, this approach only works if we have an argument that satisfies *perfect* HVZK and it turns out that this naive simulator is not guaranteed to run in expected polynomial time if our argument system is only computationally HVZK.

To make sure that our simulator V^* does indeed run in expected polynomial time, we closely follow a proof strategy due to Goldreich and Kahan [29].[7] We specify the zero-knowledge simulator Sim in Figure 2.

$\mathsf{Sim}^{\mathsf{V}^*(x,y)}(x, y)$	$\mathsf{EstimateDelta}(x, y)$
1 : $(e, d, z) \leftarrow \mathsf{Sim}'(x, y)$	$k := 0, m := 0$
2 : $d^* \leftarrow \mathsf{V}^*(e)$	**while** $k < 12\lambda$
3 : **if** $d^* = \mathsf{abort}$	$m := m + 1$
4 : **return** abort	**rewind** V^*
5 : $\widetilde{\delta} \leftarrow \mathsf{EstimateDelta}(x, y)$	$(e, d, z) \leftarrow \mathsf{Sim}'(x, y)$
6 : **for** $0 \le i < \min\left(4\lvert\mathcal{C}\rvert\lambda\widetilde{\delta}^{-1}, 2^\lambda\right)$	**if** $\mathsf{V}^*(e) \neq \mathsf{abort}$
7 : **rewind** V^*	$k = k + 1$
8 : $(e, d, z) \leftarrow \mathsf{Sim}'(x, y)$	**return** k/m
9 : $d^* \leftarrow \mathsf{V}^*(e)$	
10 : **if** $d = d^*$	
11 : **return** $\mathsf{V}^*(z)$	
12 : **return** fail	

Fig. 2: Zero-knowledge simulator for any three-move public coin honest verifier zero knowledge argument with a polynomially large challenge space.

We first observe in Lemma 2 that for any verifier V^* the probability of aborting after seeing a simulated first message output by Sim' does not differ significantly from the probability of aborting after seeing a real first protocol message.

[7] See [39] for a very nice and detailed discussion of this proof strategy.

Lemma 2. *For any polynomial time algorithm* V^* *and any* $x \in X$*, such that* $\mathcal{L}_x \neq \emptyset$ *it holds that*

$$\left| \begin{array}{r} \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = \mathsf{abort}\right] \\ -\Pr\left[(y,e,d,z) \leftarrow \textsc{Real}(x) : \mathsf{V}^*(x,y,e) = \mathsf{abort}\right] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

Proof. Let V^* be an arbitrary malicious polynomial time verifier. Consider the following distinguisher $\mathcal{D}$ against the *honest verifier* zero-knowledge property of the argument: Upon receiving (x,y) and (e,d,z) as input, the distinguisher $\mathcal{D}$ invokes V^* with fresh random coins and input (x,y,e). If V^* aborts then $\mathcal{D}$ outputs 1. Otherwise it outputs 0. We observe that $\mathcal{D}$'s distinguishing advantage against the honest verifier zero-knowledge property of the argument corresponds exactly to the difference in the abort probabilities of V^*. Since the argument is honest verifier zero-knowledge, $\mathcal{D}$'s distinguishing advantage must be negligible and Lemma 2 thus follows. □

Furthermore, we use an observation made previously by Goldreich and Kahan [29].

Lemma 3 ([29]). *For any algorithm* V^**, let* $\delta = \delta(\lambda)$ *be the probability that it does* not *abort upon seeing a simulated first message. With probability at least* $1 - 2^{-\lambda}$*, the estimate* $\widetilde{\delta}$ *in line 5 of* Sim *in Figure 2 is within a constant factor of* δ*.*

Using these two observations we will now analyze the probability that one iteration of Sim's main loop is successful. I.e. we will show that the probability that for a precomputed transcript (e,d,z), V^* upon receiving input e will return d^* with $d = d^*$ with probability at least $\delta/|\mathcal{C}| - \mathsf{negl}(\lambda)$.

Lemma 4. *Let* (P,V) *be a three-move public-coin* honest verifier *zero-knowledge argument for language* $\mathcal{L} \subseteq X \times Y$ *and let* $\mathcal{C}$ *be the associated challenge space with* $|\mathcal{C}| \leq \mathsf{poly}(\lambda)$*. Let further* V^* *be any polynomial time verifier. For any* $x \in X$*, such that* $\mathcal{L}_x \neq \emptyset$ *it then holds that*

$$\Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = d\right] \geq \frac{\delta}{|\mathcal{C}|} - \mathsf{negl}(\lambda).$$

Proof. Let V^* be an arbitrary polynomial time verifier. Now consider the following distinguisher $\mathcal{D}$ against the honest verifier zero-knowledge property of the argument. The distinguisher $\mathcal{D}$ receives as input (x,y) and (e,d,z). It initializes V^*, provides it with (x,y,e), and receives back d^*. If $d^* = \mathsf{abort}$, then $\mathcal{D}$ flips a random coin b and return that as its guess. Otherwise, $\mathcal{D}$ outputs 1 if $d = d^*$ and 0 if $d \neq d^*$. Let $\delta + \gamma$ be the probability that V^* aborts after seeing a first real message, where $|\gamma| = \mathsf{negl}(\lambda)$ by Lemma 2. By the honest verifier zero-knowledge property of the argument it must then hold that

$$\mathsf{negl}(\lambda) \geq \left| \begin{array}{r} \Pr[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathcal{D}(x,y,e,d,z) = 1] \\ -\Pr[(y,e,d,z) \leftarrow \textsc{Real}(x) : \mathcal{D}(x,y,e,d,z) = 1] \end{array} \right|$$

$$
\begin{aligned}
&\left|\begin{array}{l}
\Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathcal{D}(x,y,e,d,z)=1 \mid \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] \\
\quad \cdot \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] \\
+ \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathcal{D}(x,y,e,d,z)=1 \mid \mathsf{V}^*(x,y,e) = \mathsf{abort}\right] \\
\quad \cdot \underbrace{\Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = \mathsf{abort}\right]}_{=1-\delta} \\
= - \Pr\left[(y,e,d,z) \leftarrow \textsc{Real}(x) : \mathcal{D}(x,y,e,d,z)=1 \mid \mathsf{V}^*(x,y,e) = \mathsf{abort}\right] \\
\quad \cdot \underbrace{\Pr\left[(y,e,d,z) \leftarrow \textsc{Real}(x) : \mathsf{V}^*(x,y,e) = \mathsf{abort}\right]}_{=1-\delta-\gamma} \\
- \Pr\left[(y,e,d,z) \leftarrow \textsc{Real}(x) : \mathcal{D}(x,y,e,d,z)=1 \mid \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] \\
\quad \cdot \underbrace{\Pr\left[(y,e,d,z) \leftarrow \textsc{Real}(x) : \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right]}_{=\delta+\gamma}
\end{array}\right| \\
&= \left|\begin{array}{l}
\Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = d \mid \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] \\
\quad \cdot \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] \\
+ \underbrace{\Pr\left[b \leftarrow \{0,1\} : b=1\right]}_{=1/2} \cdot (1-\delta) - \underbrace{\Pr\left[b \leftarrow \{0,1\} : b=1\right]}_{=1/2} \cdot (1-\delta-\gamma) \\
\quad \underbrace{- \Pr\left[(y,e,d,z) \leftarrow \textsc{Real}(x) : \mathsf{V}^*(x,y,e) = d \mid \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right]}_{=1/|\mathcal{C}|} \\
\quad \cdot (\delta+\gamma)
\end{array}\right| \\
&= \left|\begin{array}{l}
\Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = d \mid \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] \\
\cdot \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] + \dfrac{\gamma}{2} - \dfrac{\delta+\gamma}{|\mathcal{C}|}
\end{array}\right| \\
&= \left|\begin{array}{l}
\Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = d \mid \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] \\
\cdot \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] - \dfrac{\delta}{|\mathcal{C}|} + \dfrac{(|\mathcal{C}|-2)\gamma}{2|\mathcal{C}|}
\end{array}\right| \\
&= \left|\begin{array}{l}
\Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = d \mid \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] \\
\cdot \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] - \dfrac{\delta}{|\mathcal{C}|} + \mathsf{negl}(\lambda)
\end{array}\right|. \qquad (1)
\end{aligned}
$$

We can now consider the two cases of the value between the absolute value bars in Equation 1 being positive, or negative. If it's positive, then it holds that

$$
\begin{aligned}
\frac{\delta}{|\mathcal{C}|} - \mathsf{negl}(\lambda) \leq &\Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = d \mid \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] \\
&\cdot \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right]. \qquad (2)
\end{aligned}
$$

If it's negative, then it must hold that

$$
\begin{aligned}
\mathsf{negl}(\lambda) \geq &- \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = d \mid \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] \\
&\cdot \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] + \frac{\delta}{|\mathcal{C}|} - \mathsf{negl}(\lambda)
\end{aligned}
$$

and thereby that

$$\frac{\delta}{|\mathcal{C}|} - \mathsf{negl}(\lambda) \leq \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = d \mid \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right] \cdot \Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) \neq \mathsf{abort}\right]. \tag{3}$$

Combining the two cases, i.e., Equations 2 and 3 we can use the law of total probability to conclude that

$$\Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = d\right] \geq \frac{\delta}{|\mathcal{C}|} - \mathsf{negl}(\lambda).$$

as claimed. □

We now want to use Lemma 4 to argue that the output of the simulator is indistinguishable from the output of V^* in a real execution. For this, consider the following. By Lemma 4, there exists a negligible function ϵ, such that

$$\Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = d\right] \geq \frac{\delta}{|\mathcal{C}|} - \epsilon(\lambda).$$

For each security parameter $\lambda \in \mathbb{N}$ we can consider two cases:

Case i. If it holds that $\delta(\lambda) > 2|\mathcal{C}|\epsilon(\lambda)$, then we have $\epsilon(\lambda) < \delta/(2|\mathcal{C}|)$ and it therefore holds that

$$\Pr\left[(y,e,d,z) \leftarrow \textsc{Simu}(x) : \mathsf{V}^*(x,y,e) = d\right] \geq \frac{\delta}{|\mathcal{C}|} - \epsilon(\lambda) > \frac{\delta}{2|\mathcal{C}|}.$$

It follows that in expectation the simulator needs at most $2|\mathcal{C}|/\delta$ rewinding attempts to obtain one non-aborting and correctly guessed execution. Via markov-inequality it follows that the probability of not having seen a single non-aborting correctly guessed execution after $4\lambda|\mathcal{C}|/\delta$ rewindings is negligible.

Lastly observe that by Lemma 3 the estimate $\widetilde{\delta}$ is within a constant factor of δ with probability $1-2^{-\lambda}$. Therefore, the simulator will output a valid transcript with a probability of $1 - \mathsf{negl}(\lambda)$, ensuring that the output of the simulator is indistinguishable from the output of V^* in a real execution with overwhelming probability.

Case ii. If it holds that $\delta(\lambda) \leq 2|\mathcal{C}|\epsilon(\lambda)$, then δ is smaller than a negligible function for this λ. Assume that in this case the rewinding strategy *always fails*. Then a real execution of V^* results in abort with probability at least $1 - (\delta - \mathsf{negl}(\lambda))$ by Lemma 2, while Sim outputs abort with probability $1 - \delta$. That means the statistical distance between the two output distributions is at most $\delta + \mathsf{negl}(\lambda)$ which is an overall negligible function. Combining the two cases, we can conclude that the distinguishing advantage against Sim is upper bounded by a negligible function for each $\lambda \in \mathbb{N}$ and thus is overall computationally indistinguishable from the output of V^* in a real execution.

It remains to bound the expected runtime of Sim. Again, by Lemma 3, the estimate $\widetilde{\delta}$ is within a constant factor of δ with probability $1-2^{-\lambda}$. But whenever the estimate is *wrong*, the runtime of the main loop is still bounded by the worst case running time of 2^λ with the simulator outputting fail. We thus have an upper bound on the expected runtime of

$$\overbrace{\mathsf{poly}(\lambda)}^{\text{lines 1-4}} + \delta\left(\overbrace{\mathsf{poly}(\lambda) + \frac{12\lambda}{\delta}\mathsf{poly}(\lambda)}^{\mathsf{EstimateDelta}} + \overbrace{\left((1-2^{-\lambda})\frac{4\lambda|\mathcal{C}|}{\widetilde{\delta}} + 2^{-\lambda}2^{\lambda}\right)\cdot\mathsf{poly}(\lambda)}^{\text{lines 6-12}}\right)$$
$$\leq \mathsf{poly}(\lambda) + (12\lambda + \mathcal{O}(1)\cdot 4\lambda|\mathcal{C}| + \delta\mathsf{poly}(\lambda))\cdot\mathsf{poly}(\lambda) \leq \mathsf{poly}(\lambda). \qquad \square$$

5 An Efficient Shuffle Argument

Let $\mathcal{C} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Rerand})$ be a perfectly and inversely rerandomizable commitment scheme with message space $\mathcal{M}$ and randomness space $\mathcal{R}$. To define shuffle arguments for $\mathcal{C}$, we first need to define partially fixable language of valid shuffles relative to a rerandomizable commitment scheme. To this end, we first define π as an algorithm that takes a vector of $\mathcal{C}$ commitments V, a permutation $p \in \mathsf{Perm}_\ell$, and randomnesses $r_0, \ldots, r_{\ell-1} \in \mathcal{R}^\ell$ as input, permutes the elements of V and randomizes each commitment. The algorithm π as well as it's inverse is described in Figure 3. We can now define the partially fixable language of valid shuffles relative to π as follows.

Definition 8 (Valid Shuffle). *Let $\mathcal{C}$ be a perfectly rerandomizable commitment scheme with commitment space $\mathcal{C}$. The language $\mathsf{Shuffle}_\ell \subseteq \mathcal{C}^\ell \times \mathcal{C}^\ell$ of valid shuffles of vectors of length ℓ is defined as*

$$\mathsf{Shuffle}_\ell = \left\{(V, V') \in \mathcal{C}^\ell \times \mathcal{C}^\ell \;\middle|\; \exists\,(p, \vec{r}) \in \mathsf{Perm}_\ell \times \mathcal{R}^\ell.\; V' = \pi(V, p, \vec{r})\right\}$$

Shuffles are transitive as stated by the following lemma.

Lemma 5. *If $(V, V') \in \mathsf{Shuffle}_\ell$ and $(V', V'') \in \mathsf{Shuffle}_\ell$, then $(V, V'') \in \mathsf{Shuffle}_\ell$.*

Proof. Since permutations are closed under composition and since, by assumption on the commitment scheme, it holds that for any $r_1, r_2 \in \mathcal{R}$, there exists an $r_3 \in \mathcal{R}$ such that $\mathsf{Rerand}(\mathsf{ck}, \mathsf{Rerand}(\mathsf{ck}, c, r_1), r_2) = \mathsf{Rerand}(\mathsf{ck}, c, r_3)$, the lemma immediately follows. $\square$

$\pi(V, p, r_0, \ldots, r_{\ell-1})$

for $0 \leq i < \ell$
  $V'[p(i)] := \mathsf{Rerand}(\mathsf{ck}, V[i], r_{p(i)})$
return V'

$\pi^{-1}(V', p, r_0, \ldots, r_{\ell-1})$

for $0 \leq i < \ell$
  $V[i] := \mathsf{Rerand}^{-1}(\mathsf{ck}, V'[p(i)], r_i)$
return V'

Fig. 3: The algorithms for rerandomizing and permuting a vector of ciphertexts, as well as its inverse.

Definition 9 (Shuffle Argument). *An interactive arguments for partially fixed statements of* $\mathsf{Shuffle}_\ell$ *relative to any perfectly rerandomizable commitment scheme* $\mathcal{C}$ *is called a shuffle argument for* $\mathcal{C}$*.*

Now, let $(\mathcal{F}, \mathsf{Puncture})$ be a puncturable pseudorandom function with key length $k(\lambda)$, input length $i(\lambda)$, and range $\mathsf{Perm}_\ell \times \mathcal{R}^\ell$. Let H be a collision resistant hash function. In Figure 4 we then describe a simple three-move shuffle argument. We will first prove that this protocol is a zero knowledge shuffle argument as

$\mathsf{P}_0(V_0)$	$\mathsf{V}_1(V'_0, V'_t, c)$	$\mathsf{V}_2(k')$
$k \leftarrow \{0,1\}^{k(\lambda)}$	$d \leftarrow \{1, \ldots, t\}$	**for** $1 \leq i < d$
for $1 \leq i \leq t$	**return** d	$\quad V'_i := \pi(V'_{i-1}, \mathcal{F}(k', i))$
$\quad V_i := \pi(V_{i-1}, \mathcal{F}(k, i))$		**for** $t > i \geq d$
$c := \mathsf{H}(V_1, \ldots, V_{t-1})$	$\mathsf{P}_2(d)$	$\quad V'_i := \pi^{-1}(V'_{i+1}, \mathcal{F}(k', i+1))$
return $(V_t, (c, k))$	**if** $d \in \{1, \ldots, t\}$	**if** $\mathsf{H}(V'_1, \ldots, V'_{t-1}) = c$
	$\quad k' \leftarrow \mathsf{Puncture}(k, d)$	$\quad$ **return** 1
$\mathsf{P}_1(V_0, (c, k))$	$\quad$ **return** k'	**else return** 0
return c	**else**	
	$\quad$ **return** $\perp$	

Fig. 4: An algorithmic description of the shuffle argument.

stated in the following theorem.

Theorem 6. *Let* $\mathcal{C} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Rerand})$ *be a perfectly and inversely rerandomizable commitment scheme with message space* $\mathcal{M}$ *and randomness space* $\mathcal{R}$*. Let* $(\mathcal{F}, \mathsf{Puncture})$ *be a puncturable pseudorandom function with key length* $k(\lambda)$*, input length* $i(\lambda)$*, and range* $\mathsf{Perm}_\ell \times \mathcal{R}^\ell$*. Let* H *be a collision resistant hash function. Then the argument system* $\langle \mathsf{P} = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2) \rangle$ *described in Figure 4 is a zero-knowledge shuffle argument with soundness error* $1/t$ *for* $\mathcal{C}$*.*

Theorem 6 follows from Lemmas 7 and 10 as well as Corollary 12, which we prove in the following.

Lemma 7. *Let* $\mathcal{C}$*,* $(\mathcal{F}, \mathsf{Puncture})$*, and* H *be as in Theorem 6. Then the argument system described in Figure 4 is complete.*

Proof. We need to show that it always holds that $(V_0, V'_t) \in \mathsf{Shuffle}_\ell$ and that, in an interaction with the honest prover, the verifier always accepts and outputs 1.

Claim 8. *For any* $V_0 \in \mathcal{C}^\ell$ *and any* $(V_t, (c, k)) \leftarrow \mathsf{P}_0(V_0)$ *it holds that* $(V_0, V'_t) \in \mathsf{Shuffle}_\ell$*.*

The prover computes each V_i as $V_i := \pi(V_{i-1}, \mathcal{F}(k,i))$, where $\mathcal{F}(k,i)$ outputs the description of a permutation and ℓ random values in $\mathcal{R}$. It then follows from the definition of π in Figure 3 and the perfect rerandomizability of $\mathcal{C}$, that for $0 < i \leq t$, $(V_{i-1}, V_i) \in \mathsf{Shuffle}_\ell$. Using Lemma 5 we can then conclude by induction, that $(V_0, V_t) \in \mathsf{Shuffle}_\ell$.

Claim 9. *For any $V_0 \in \mathcal{C}^\ell$ and any $(V_t, (c,k)) \leftarrow \mathsf{P}_0(V_0)$ any honest execution of $\langle \mathsf{P}(V_0, V_t, (c,k)), \mathsf{V}(V_0, V_t)\rangle$ will always output 1.*

We note that V_2 uses k' to recompute all V'_i for $1 \leq i < t$. For $1 \leq i < d$, this happens by computing $V'_i := \pi(V'_{i-1}, \mathcal{F}(k',i)) = \pi(V'_{i-1}, \mathcal{F}(k,i))$, where the last equality holds by the functionality requirement of the puncturable PRF, since $i \neq d$. As it always holds that $V'_0 = V_0$, it follows by induction over i, that $V'_i = V_i$ for $1 \leq i < d$.

For $d \leq i < t$, the verifier computes $V'_i := \pi^{-1}(V'_{i+1}, \mathcal{F}(k', i+1))$. It always holds that $V_t = V'_t$ and the prover computes $V_t := \pi(V_{t-1}, \mathcal{F}(k,t))$. This gives us $V'_{t-1} = \pi^{-1}(\pi(V_{t-1}, \mathcal{F}(k,t)), \mathcal{F}(k,t))$, since $t \neq d$. By the definition of π and π^{-1} we thus have that for $0 \leq j < \ell$ it holds that

$$V'_{t-1}[j] = \mathsf{Rerand}^{-1}(\mathsf{Rerand}(V'[j], r_j), r_j) = V_{t-1}[j]$$

for some value of r_j. The last equality follows from the inverse rerandomizability of $\mathcal{C}$. Therefore, it follows that $V'_{t-1} = V_{t-1}$ and by induction that $V'_i = V_i$ for $d \leq i < t$.

We thus have that with probability 1, $(V'_1, \ldots, V'_{t-1}) = (V_1, \ldots, V_{t-1})$ and therefore $\mathsf{H}(V'_1, \ldots, V'_{t-1}) = c$. It thus follows that V_2 outputs 1.

Combining the two claims Lemma 7 immediately follows. □

Lemma 10. *Let $\mathcal{C}$, $(\mathcal{F}, \mathsf{Puncture})$, and H be as in Theorem 6. Then the argument system described in Figure 4 is sound with soundness error $1/t$.*

Proof. Let $(V_0^*, V_t^*) \notin \mathsf{Shuffle}_\ell$ and let P^* be an arbitrary probabilistic polynomial time prover. We will show, that

$$\Pr[b \leftarrow \langle \mathsf{P}^*(1^\lambda), \mathsf{V}(x,y)\rangle : b = 1] \leq 1/t + \mathsf{negl}(\lambda).$$

We will assume without loss of generality, that P^* actually sends a first message c and that c is fixed.[8]

Let $d_0, d_1 \in \{1, \ldots, t\}$ be two arbitrary distinct challenges and let $k'_0 \leftarrow \mathsf{P}^*(d_0)$ and $k'_1 \leftarrow \mathsf{P}^*(d_1)$ be the corresponding responses. Consider, that the verifier works by recomputing $\mathbf{V}_b = (V_1^b, \ldots, V_{t-1}^b)$ and checking that it hashes to c. The verifier computes V_i^b as $V_i^b = \pi(V_{i-1}^b, \mathcal{F}(k'_b, i))$ for $i < d_b$ and as $\pi^{-1}(V_{i+1}^b, \mathcal{F}(k'_b, i+1))$ for $i \geq d_b$.

By definition of π and π^{-1}, this implies for $i < d_0$ that $(V_{i-1}^0, V_i^0) \in \mathsf{Shuffle}_\ell$ and for $i \geq d_0$ that $(V_i^0, V_{i+1}^0) \in \mathsf{Shuffle}_\ell$. By Lemma 5 we can thus conclude that

$$(V_0, V_{d_0-1}^0) \in \mathsf{Shuffle}_\ell \quad \text{and} \quad (V_{d_0}^0, V_t) \in \mathsf{Shuffle}_\ell. \tag{4}$$

[8] This is without loss of generality, since we can fix the prover's random coins to those that lead to the highest success probability.

Since $d_1 \neq d_0$, we have by the same reasoning, that

$$(V^1_{d_0-1}, V^1_{d_0}) \in \mathsf{Shuffle}_\ell. \tag{5}$$

If it were true, that $(V^0_{d_0-1}, V^0_{d_0}) = (V^1_{d_0-1}, V^1_{d_0})$ then it would follow from Equations 4 and 5 by Lemma 5 that $(V_0, V_t) \in \mathsf{Shuffle}_\ell$ which would contradict the initial assumption. Therefore, it must hold that $(V^0_{d_0-1}, V^0_{d_0}) \neq (V^1_{d_0-1}, V^1_{d_0})$ and $\mathbf{V}_0 \neq \mathbf{V}_1$. This would mean, however, that we could break collision resistance of H by presenting $\mathbf{V}_0, \mathbf{V}_1$ with probability

$$\Pr[(k'_0, r_0) \leftarrow \mathsf{P}^*(d_0) : \mathsf{V}_2(k'_0) = 1] \cdot \Pr[(k'_1, r_1) \leftarrow \mathsf{P}^*(d_1) : \mathsf{V}_2(k'_1) = 1].$$

Since the hash function is collision resistant, it follows that the above probability can be bounded by a negligible function. Thus, at least one of the two probabilities must be itself negligible. Since we have shown the above for all pairs of distinct challenges, this means that there can exist at most one challenge $d \in \{1, \ldots, t\}$ such that $\Pr[(k', r) \leftarrow \mathsf{P}^*(d) : \mathsf{V}_2(k') = 1]$ is non-negligible. It thus ultimately follows that

$$\begin{aligned}
&\Pr[b \leftarrow \langle \mathsf{P}^*(1^\lambda), \mathsf{V}(x, y)\rangle : b = 1] \\
=&\sum_{d=1}^{t} \Pr[\mathsf{V}_1(c) = d] \cdot \Pr[k' \leftarrow \mathsf{P}^*(d) : \mathsf{V}_2(k') = 1] \\
=&\frac{1}{t} \cdot \sum_{d=1}^{t} \Pr[k' \leftarrow \mathsf{P}^*(d) : \mathsf{V}_2(k') = 1] \\
\leq&\frac{1}{t} \cdot (1 + (t-1) \cdot \mathsf{negl}(\lambda)) \leq \frac{1}{t} + \mathsf{negl}(\lambda)
\end{aligned}$$

as claimed.

Lemma 11. *Let $\mathcal{C}$, $(\mathcal{F}, \mathsf{Puncture})$, and H be as in Theorem 6. Then the argument system described in Figure 4 is honest verifier zero-knowledge.*

Before we prove Lemma 11, we first state the following simple corollary which follows immediately from by combining Lemma 11 with Theorem 1.

Corollary 12. *Let $\mathcal{C}$, $(\mathcal{F}, \mathsf{Puncture})$, and H be as in Theorem 6. Then the argument system described in Figure 4 is zero-knowledge.*

Proof (Lemma 11). By Definition 7 we need to show that there exists a simulator Sim, such that

$$\left| \begin{aligned} &\Pr[(V_t, \mathcal{T}) \leftarrow \textsc{Simu}(V_0) : \mathcal{D}(V_0, V_t, \mathcal{T}) = 1] \\ -&\Pr[(V_t, \mathcal{T}) \leftarrow \textsc{Real}(V_0) : \mathcal{D}(V_0, V_t, \mathcal{T}) = 1] \end{aligned} \right| \leq \mathsf{negl}(\lambda). \tag{6}$$

We specify the honest-verifier zero-knowledge simulator in Figure 5 and use a series of game hops specified in Figures 6 and 7 to prove that the above equation holds.

$\mathsf{Sim}(V_0, V_t)$

$d \leftarrow \{1, \ldots, t\}$
$k \leftarrow \{0,1\}^{k(\lambda)}$
for $1 \leq i < d$
  $V_i := \pi(V_{i-1}, \mathcal{F}(k, i))$
for $t > i \geq d$
  $V_i := \pi^{-1}(V_{i+1}, \mathcal{F}(k, i+1))$
$c := \mathsf{H}(V_1, \ldots, V_{t-1})$
$k' \leftarrow \mathsf{Puncture}(k, d)$
return $(c; d; k')$

Fig. 5: Honest-verifier zero knowledge simulator for the three-move protocol specified in Figure 4

$\mathsf{Game}_1(V_0)$

$d \leftarrow \{1, \ldots, t\}$
$k \leftarrow \{0,1\}^{k(\lambda)}$
for $1 \leq i < d$
  $V_i := \pi(V_{i-1}, \mathcal{F}(k, i))$
$p \leftarrow \mathsf{Perm}_\ell, \vec{r} \leftarrow \mathcal{R}^\ell$
$V_t := \pi(V_0, p, \vec{r})$
for $t > i \geq d$
  $V_i := \pi^{-1}(V_{i+1}, \mathcal{F}(k, i+1))$
$c := \mathsf{H}(V_1, \ldots, V_{t-1})$
$k' \leftarrow \mathsf{Puncture}(k, d)$
$b \leftarrow \mathcal{D}(V_0, V_t, (c; d; k')$
return b

Fig. 6: Game 1 for the proof of honest verifier zero-knowledge.

We first observe that

$$\Pr[(V_t, \mathcal{T}) \leftarrow \textsc{Simu}(V_0) : \mathcal{D}(V_0, V_t, \mathcal{T}) = 1] = \Pr[Game_1(V_0) = 1] \tag{7}$$

This is easily verified. V_t is chosen uniformly at random from all valid shufflings in both cases. Further, c, d and k' are all computed in exactly the same way. Similarly, we observe that

$$\Pr[(V_t, \mathcal{T}) \leftarrow \textsc{Real}(V_0) : \mathcal{D}(V_0, V_t, \mathcal{T}) = 1] = \Pr[Game_4(V_0)] \tag{8}$$

This is also easily verified. $Game_4$ computes V_t and (c, k) in exactly the same way as P_0, lets the honest verifier $\mathsf{V}_1(V_0, V_t, c)$ choose d and finally computes k' in exactly the same manner as P_2. What remains is to bound the differences between each pair of consecutive games.

Hop from $Game_1$ to $Game_2$. The changes between the two games are purely syntactic. In $Game_1$ the final shuffling V_t is sampled uniformly at random from all valid shuffles of V_0. In $Game_2$ the final shuffling V_t is computed as the composition of several intermediate *valid* shuffles. The shuffling at position d is chosen uniformly at random and *independently* from all other shuffles. Since all previous shuffles are valid, this makes V_d a uniformly random shuffling of V_0. Further, since all following shuffles are valid and the shuffling of V_d was independent, this makes V_t a uniformly random shuffling of V_0. Therefore V_t is distributed identically in both games. By the perfect and inverse rerandomizability of $\mathcal{C}$, it

$\mathsf{Game}_2(V_0)$	$\mathsf{Game}_3(V_0)$	$\mathsf{Game}_4(V_0)$
$d \leftarrow \{1,\ldots,t\}$	$d \leftarrow \{1,\ldots,t\}$	
$k \leftarrow \{0,1\}^{k(\lambda)}$	$k \leftarrow \{0,1\}^{k(\lambda)}$	$k \leftarrow \{0,1\}^{k(\lambda)}$
for $1 \leq i < d$	**for** $1 \leq i \leq t$	**for** $1 \leq i \leq t$
$V_i := \pi(V_{i-1}, \mathcal{F}(k,i))$	$V_i := \pi(V_{i-1}, \mathcal{F}(k,i))$	$V_i := \pi(V_{i-1}, \mathcal{F}(k,i))$
$p \leftarrow \mathsf{Perm}_\ell, \vec{r} \leftarrow \mathcal{R}^\ell$		
$V_d := \pi(V_{d-1}, p, \vec{r})$		
for $d < i \leq t$		
$V_i := \pi(V_{i-1}, \mathcal{F}(k,i))$		
$c := \mathsf{H}(V_1,\ldots,V_{t-1})$	$c := \mathsf{H}(V_1,\ldots,V_{t-1})$	$c := \mathsf{H}(V_1,\ldots,V_{t-1})$
		$d \leftarrow \mathsf{V}_1(V_0, V_t, c)$
$k' \leftarrow \mathsf{Puncture}(k,d)$	$k' \leftarrow \mathsf{Puncture}(k,d)$	$k' \leftarrow \mathsf{Puncture}(k,d)$
$b \leftarrow \mathcal{D}(V_0, V_t, (c;d;k'))$	$b \leftarrow \mathcal{D}(V_0, V_t, (c;d;k'))$	$b \leftarrow \mathcal{D}(V_0, V_t, (c;d;k'))$
return b	**return** b	**return** b

Fig. 7: Game 2 through 4 for the proof of honest verifier zero-knowledge.

makes no difference, whether the V_i for $d < i < t$ are computed in the "forward" direction from V_{i-1} or in the "backwards" direction from V_{i+1}. Therefore the two games are perfectly equivalent, and it holds that

$$\Pr[Game_1(V_0) = 1] = \Pr[Game_2(V_0) = 1]. \tag{9}$$

Hop from $Game_2$ to $Game_3$. Note that the only difference between the two games is in the computation of V_d, which is computed as a uniformly random shuffle in $Game_2$ and as a *pseudorandom* shuffle in $Game_3$. This means we can bound the difference between the two games using a reduction to the pseudorandomness of the puncturable pseudorandom function. Specifically we use $\mathcal{D}$ as a distinguisher against $\mathcal{F}$ by requesting a key punctured on d and after receiving k' and $y = (p, \vec{r})$, computing V_i with key k' as in $Game_2$ except that we compute V_d using $y = (p, \vec{r})$. It is easy to see, that if y is uniformly random, then we perfectly simulate $Game_2$, whereas if $y = \mathcal{F}(k,d)$ we perfectly simulate $Game_3$. By the security of $\mathcal{F}$ it must therefore hold that

$$|\Pr[Game_2(V_0) = 1] - \Pr[Game_3(V_0) = 1]| \leq \mathsf{negl}(\lambda) \tag{10}$$

Hop from $Game_3$ to $Game_4$. The changes between the two games are again merely syntactic. In particular, the games behave identically, except that $Game_3$ chooses d uniformly at random from $\{1,\ldots,t\}$ whereas $Game_4$ lets the verifier choose $d \leftarrow \mathsf{V}_0(V_0, V_t, c)$. However, by definition of V_0 these two sampling strategies are identical and it holds that

$$\Pr[Game_3(V_0) = 1] = \Pr[Game_4(V_0) = 1]. \tag{11}$$

Finally, combining Equations 9 through 11, we get

$$|\Pr[Game_1(V_0) = 1] - \Pr Game_4(V_0) = 1| \leq \mathsf{negl}(\lambda), \tag{12}$$

which combined with Equation 7 and Equation 8 gives us Equation 6 thus concluding the proof. □

In the full version of this paper, we additionally show how to modify the constrution to achieve straightline simulation in strict polynomial time in the CRS model. We also show how the amortized efficiency of the construction can be improved through batching and how the verifier's computational overhead can be optimized at the cost of a slightly worse communication complexity.

6 Public Accountability

If a public-key infrastructure (PKI), which associates the prover with a public key pk, is available, then we can discourage malicious provers from attempting to cheat by ensuring that the verifier obtains a publicly verifiable certificate that attests any failed cheating attempt by the prover. In the context of blockchains, such a certificate could for example be used to punish the prover through financial penalties.

In the following definition, we define this property by requiring the existence of a judge algorithm that can verify valid certificates and cannot be fooled by invalid certificates that falsely accuse an honest prover of misbehavior. For the sake of readability, we implicitly assume that the verifier has access to the prover's public key amd the prover has access to their own public and secret keys.

Definition 10 (Publicly-Accountable Zero-Knowledge Arguments). *Let $\mathcal{L} \subseteq X \times Y$ be a partially fixable language. Let (P, V) be an interactive zero-knowledge argument for $\mathcal{L}$ with soundness error ϵ in the PKI model. We say that the argument system is publicly accountable, if there exists a PPT algorithm Setup and a deterministic polynomial time judge algorithm J, such that the following conditions hold:*

Accountability: *Fix any $(x, y) \notin \mathcal{L}$ and let P^* be a malicious probabilistic polynomial time prover with*

$$\Pr[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda); b \leftarrow \langle \mathsf{P}^*(\mathsf{crs}), \mathsf{V}(\mathsf{crs}, x, y)\rangle : b = 1] \geq \delta\epsilon,$$

where the probability is taken over the random coins of the prover and the verifier and $0 < \delta \leq 1$. Then it holds that

$$\begin{aligned}&\Pr[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda); \mathsf{cert} \leftarrow \langle \mathsf{P}^*(\mathsf{crs}), \mathsf{V}(\mathsf{crs}, x, y)\rangle : \mathsf{J}(\mathsf{crs}, \mathsf{pk}, \mathsf{cert}) = 1] \\ \geq &\delta(1 - \epsilon) - \mathsf{negl}(\lambda),\end{aligned}$$

where the probability is taken over the random coins of the prover and the verifier.

Defamation-Freeness: *For any $x \in X$ with $\mathcal{L}_x \neq \emptyset$, for any honest prover P and malicious probabilistic polynomial time verifier V^*, it holds that*

$$\Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda); \\ (y, \mathsf{aux}) \leftarrow \mathsf{P}_0(\mathsf{crs}, x); \\ \mathsf{cert} \leftarrow \langle \mathsf{P}_1(x, y, \mathsf{aux}), \mathsf{V}^*(\mathsf{crs}, x, y)\rangle \end{array} : \mathsf{J}(\mathsf{pk}, \mathsf{cert}) = 1\right] \leq \mathsf{negl}(\lambda).$$

We show that the three move zero-knowledge shuffle argument (P, V) described in Figure 4 in Section 5 can be transformed into a publicly-accountable zero-knowledge argument.

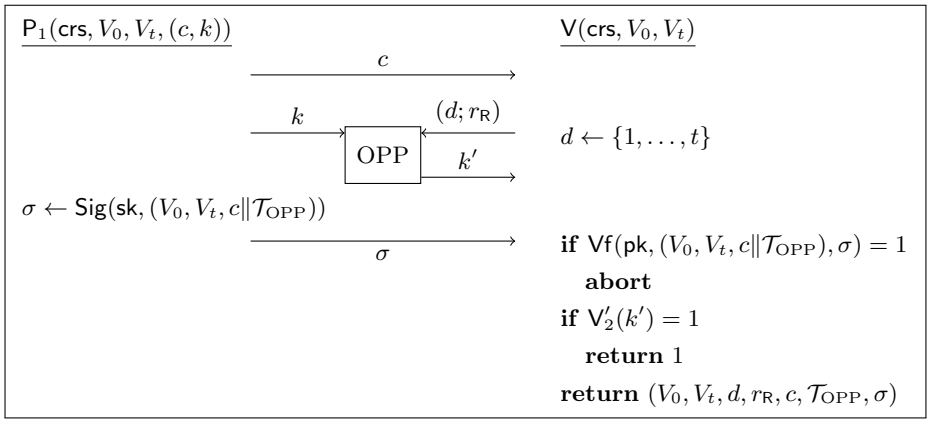

Fig. 8: The publicly-accountable transformation of the three-move shuffle argument. Here r_R refers to the random tape V uses to execute R in the OPP. $\mathcal{T}_{\mathrm{OPP}}$ refers to the full transcript resulting from the OPP execution.

Theorem 13. *Let $\langle \mathsf{P}', \mathsf{V}' \rangle$ be the three-move zero-knowledge shuffle argument described in Figure 4 in Section 5. Let (S, R) be a secure receiver-extractable, oblivious key puncturing protocol for the puncturable PRF used in $\langle \mathsf{P}', \mathsf{V}' \rangle$. Let $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Vf})$ be an existentially unforgeable signature scheme. Then the protocol $\langle \mathsf{P}, \mathsf{V} \rangle$ with $\mathsf{P} = (\mathsf{P}'_0, \mathsf{P}_1)$, with P_1 and V as specified in Figure 8 is a publicly-accountable zero-knowledge argument with soundness error $1/t$.*

Proof. We now show that this construction is indeed a publicly-accountable zero-knowledge argument.

Lemma 14. *The argument system (P, V) is complete.*

Proof. This directly follows from the completeness of the underlying argument system $(\mathsf{P}', \mathsf{V}')$, the completeness of the oblivious key puncturing protocol, and the correctness of the signature scheme. □

Lemma 15. *The argument system (P, V) is sound with soundness error $1/t$.*

Proof. Let $(V_0, V_t) \notin \mathsf{Shuffle}_\ell$ and let P^* be an arbitrary malicious probabilistic polynomial time prover against (P, V). We use P^* to construct a probabilistic polynomial time prover $\widetilde{\mathsf{P}}$ against the original three move argument system $(\mathsf{P}', \mathsf{V}')$ as follows. $\widetilde{\mathsf{P}}$ initializes P^* with uniform random coins. Without loss of generality assume that P^* outputs a first message c, which $\widetilde{\mathsf{P}}$ forwards to the verifier. The verifier outputs a challenge d. $\widetilde{\mathsf{P}}$, acting on behalf of a simulated verifier towards P^*, engages in an execution of the oblivious puncturing protocol as the receiver, where d is the choice index. Let k' be the output that $\widetilde{\mathsf{P}}$ receives from P^* in this execution. $\widetilde{\mathsf{P}}$ forwards k' to the verifier. It is straightforward to see that $\widetilde{\mathsf{P}}$'s success probability is at least as large as the success probability of P^*. Since by Theorem 6 $\widetilde{\mathsf{P}}$'s success probability is bounded by $1/t + \mathsf{negl}(\lambda)$ the lemma follows. □

Lemma 16. *The argument system* (P, V) *is zero-knowledge.*

Proof. Let V^* be an arbitrary verifier to which we have blackbox access. Let $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ be the simulator from the sender simulation property of the OPP. We construct a verifier $\widetilde{\mathsf{V}}'$ for the underlying argument system, which makes blackbox use of V^* and works as follows. First we let $\widetilde{\mathsf{V}}'$ sample a signature key pair $(\mathsf{sk}, \mathsf{pk})$ and whenever V^* asks the (simulated) PKI for the verification key of the prover, our verifier will return pk. Then $\widetilde{\mathsf{V}}'$ generates a simulated common reference string for the OPP as $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda)$ and initializes $\mathsf{V}^*(\mathsf{crs}, 1^\lambda)$. Upon receiving a first prover message c, verifier $\widetilde{\mathsf{V}}'$ forwards the message to V^*. Then, $\widetilde{\mathsf{V}}'$ uses $\mathsf{Sim}_1(\mathsf{crs}, \mathsf{td})$ to execute the oblivious key puncturing protocol with V^*. If and when Sim_1 makes its single query d to its puncturing oracle, $\widetilde{\mathsf{V}}'$ simulates the oracle by outputting d as its challenge message and returning the response k' as the answer. Finally, $\widetilde{\mathsf{V}}'$ outputs whatever V^* outputs. Due to the sender simulation property of the OPP, the view of V^* when simulated by $\widetilde{\mathsf{V}}'$ is computationally indistinguishable from a real execution. Therefore the output of $\widetilde{\mathsf{V}}'$ is computationally indistinguishable from the output of V^* in a real execution of the publically accountable protocol. Since, $\widetilde{\mathsf{V}}'$ is a verifier for the underlying three-move argument and by Corollary 12 that argument is zero-knowledge, there exists a zero-knowledge simulator Sim' that can simulate this output given only blackbox access to $\widetilde{\mathsf{V}}'$. We can, thus, define the zero-knowledge simulator Sim with blackbox access to V^* simply as Sim' with blackbox access to $\widetilde{\mathsf{V}}'$. □

Lemma 17. *The argument system* (P, V) *satisfies accountability.*

Proof. The judge algorithm J is specified in Figure 9. Let $(V_0, V_t) \notin \mathsf{Shuffle}_\ell$, let P^* be an arbitrary malicious PPT prover. Let $\mathsf{noAbort}(\vec{r})$ be the event that P^* does not abort and sends a valid signature when the parties run on random tapes $\vec{r} = (r_\mathsf{P}, r_\mathsf{V})$. Let $d(\vec{r})$ be the function that returns the verifier's challenge. Without loss of generality, we assume that for any fixed first message of the prover, there exists a set of verifier challenges $D(\vec{r})$, which the prover successfully answers in such a way that the verifier accepts.

$\mathsf{J}(\mathsf{crs}, \mathsf{pk}, \mathsf{cert} = (V_0, V_t, d, r_\mathsf{R}, c, \mathcal{T}, \sigma))$

if $\mathsf{Vf}(\mathsf{pk}, (V_0, V_t, c \| \mathcal{T}), \sigma) = 0$
  return 0
$k' := \mathsf{TCheck}(\mathsf{crs}, \mathcal{T}, d, r_\mathsf{R})$
if $k' = \bot$
  return 0
$b := \mathsf{V}'_2(k')$
return $b \oplus 1$

Fig. 9: The judge algorithm J for the publicly-accountable transformation of a three-move zero-knowledge argument. Here TCheck refers to an algorithm that given a CRS, an OPP transcript $\mathcal{T}$, an input d and a random tape r_R outputs R's output, if the transcript is consistent with d and r_R and $\bot$ otherwise.

From the receiver privacy of the OPP, it follows that

$$\left|\Pr[\mathsf{noAbort}(\vec{r}) \mid d(\vec{r}) \in D(\vec{r})] - \Pr[\mathsf{noAbort}(\vec{r}) \mid d(\vec{r}) \notin D(\vec{r})]\right| \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the random coins $\vec{r}$ of the parties. We observe that

$$\begin{aligned}
&\Pr[\mathsf{cert} \leftarrow \langle \mathsf{P}^*(1^\lambda; r_\mathsf{P}), \mathsf{V}(x, y; r_\mathsf{V})\rangle : \mathsf{J}(\mathsf{pk}, \mathsf{cert}) = 1 \mid (V_0, V_t) \notin \mathsf{Shuffle}_\ell] \\
\geq\ &\Pr[\mathsf{noAbort}(\vec{r}) \mid d(\vec{r}) \notin D(\vec{r})] \cdot \Pr[d(\vec{r}) \notin D(\vec{r})] \\
\geq\ &(\Pr[\mathsf{noAbort}(\vec{r}) \mid d(\vec{r}) \in D(\vec{r})] - \mathsf{negl}(\lambda)) \cdot \Pr[d(\vec{r}) \notin D(\vec{r})] \\
\geq\ &(\delta - \mathsf{negl}(\lambda)) \cdot (1 - \epsilon - \mathsf{negl}(\lambda)) \\
\geq\ &\delta(1 - \epsilon) - \mathsf{negl}(\lambda)
\end{aligned}$$

□

Lemma 18. *The argument system* (P, V) *is defamation-free.*

Proof. Let $(V_0, V_t, d, r_\mathsf{R}, c, \mathcal{T}, \sigma))$ be the certificate presented by V^*. We observe, that the existential unforgeability of the signature scheme implies that except with negligible probability, c and $\mathcal{T}$ must have originated from an interaction with the honest prover on input (V_0, V_t). The completeness of the OPP implies that TCheck will either output $\bot$, in which case J will output 0, or the correct response k' to the challenge d. It follows that (c, d, k') is the view of V' in an honest execution of the three-move shuffle argument. If $(V_0, V_t) \in \mathsf{Shuffle}_\ell$, the completeness of the argument implies that have $\mathsf{V}'_2(k') = 1$ and J will output 0. The lemma directly follows from the above observations. □

The theorem statement follows from Lemmas 15, 16, 17, and 18. □

7 Instantiation and Comparison

To evaluate the practical usefulness of the shuffle argument from Section 5 and its publicly accountable counterpart from Section 6, we explore how to instantiate

them in practice. We are particularly interested in the concrete communication complexities of such instantiations, since this is where our constructions shine. Towards this goal, we need to pick specific instantiations of the underlying building blocks, such as the collision-resistant hash function and the PPRF with its oblivious puncturing protocol. We aim for a security level of roughly 128 bits.

Scheme	Assumptions	Trusted Setup	Soundness	Communication Cost (byte)
This Work	CRHF,PRG	None	2^{-2}	81
			2^{-5}	153
			$2^{-\gamma}$	$32 + \lceil \gamma \cdot 24\frac{1}{8} \rceil$
This Work (with accountability)	CRHF,PRG,DDH	CRS	2^{-2}	416
			2^{-5}	992
			$2^{-\gamma}$	$32 + \gamma \cdot 192$
This Work (with accountability)	CRHF,PRG,RLWE	CRS	2^{-2}	$0.44 \cdot 2^{20}$
			2^{-5}	$1.1 \cdot 2^{20}$
			$2^{-\gamma}$	$32 + \gamma \cdot 0.215 \cdot 2^{20}$
Bayer-Groth [9]	Discrete Logarithm	CRS	$\mathsf{negl}(\lambda)$	700,000
Bulletproofs [17]	Discrete Logarithm	CRS	$\mathsf{negl}(\lambda)$	1,600
SNARKs [32]	Generic Bilinear Group	SRS	$\mathsf{negl}(\lambda)$	144

Table 1: Transcript size of shuffle arguments for vectors of length 100,000. The reported numbers for our constructions correspond to the instantiations described in Section 7 and are independent of the vector length and the type of commitment being shuffled. The numbers for Bayer-Groth [9] are taken from their paper for an instantiation in a q order subgroup of $\mathbb{Z}_p^*$ with $|q| = 160$ and $|p| = 1024$ shuffling ElGamal ciphertexts. For Bulletproofs [17], we consider an instantiation in ristretto255 [48] for shuffling Pedersen commitments. For the Groth SNARK [32] we consider an instantiation with curve BLS12-381 [13] and observe that the numbers are independent of the kind of commitments being shuffled and the vector length.

The Hash Function The collision resistant hash function can in practice be instantiated using any of SHA-256 [46], SHA3-256, or SHAKE256 [47] with 256 bit output length. Any of these instantiations leads to hashes of size $|c| = 256$ bits.

The Puncturable PRF The PPRF can be instantiated using the construction of Goldreich, Goldwasser and Micali (GGM) [28]. This construction relies on an internal length-doubling pseudorandom generator which can be instantiated using a secure stream cipher, such as AES [4] in CTR mode or ChaCha20 [11].

Taking the losses introduced by both the security proof of GGM as well as the proof of zero-knowledge into account, we require a PRG with $128 + \lceil 2 \log t + \log \log t \rceil$ bits of security to achieve our security goal of 128 bits. For reasonable values of t, using AES-192 in CTR mode would thus suffice. Simply instantiating GGM like this yields a PPRF with range $\{0,1\}^{192}$. As explained in Section 2, to construct a PPRF with range $\mathsf{Perm}_\ell \times \mathcal{R}^\ell$, the output can be stretched using a PRG and combined with the Fisher-Yates shuffle [24] by using the stretched output as the random tape of the shuffling algorithm. The necessary PRG can again be instantiated using, e.g., AES-192 in CTR mode or ChaCha20. Overall, the size of a punctured key with the AES-192 instantiation is $|k'| = \lceil \log t \rceil \cdot 192$ bits.

The Oblivious Puncturing Protocol Using the PPRF construction of GGM mentioned above, we can use the oblivious puncturing protocol described in [14], which itself relies on $\lceil \log t \rceil$ many oblivious transfers with active security. These can be instantiated with the 2-round UC secure protocol of Peikert, Vaikunanthan, and Waters [44] over ristretto255 [48] by relying on the DDH assumption. From a computational point of view, the parties need to perform $10\lceil \log t \rceil$ exponentiations, $6\lceil \log t \rceil$ multiplications, and $2t$ evaluations of a PRG for one oblivious puncturing. Using our instantiation, the OPP runs in two rounds and thus the overall protocol runs in three, since the signature σ can be sent in parallel with the second message of the OPP from the prover to the verifier.

To obtain post-quantum security, we can instantiate the oblivious transfer with the protocol of Micciancio and Sorrell [41], which relies on the ring learning with errors assumption.

Acknowledgments

The authors would like to thank the anonymous PKC reviewer for pointing out the efficient OPP instantiation implicit in [14] as a practical replacement for general purpose PIR. The authors would also like to thank Ivan Damgård for the insightful discussions about the Goldreich and Kahan proof technique as well as Diego F. Aranha and Cathie Yun for information about practical instantiations of SNARKs and Bulletproofs respectively.

References

1. Abe, M.: Universally verifiable mix-net with verification work independent of the number of mix-servers. In: EUROCRYPT'98. pp. 437–447. LNCS (1998)
2. Abe, M.: Mix-networks on permutation networks. In: ASIACRYPT'99. pp. 258–273. LNCS (1999)
3. Abe, M., Hoshino, F.: Remarks on mix-network based on permutation networks. In: PKC 2001. pp. 317–324. LNCS (2001)
4. Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce (Nov 2001)

5. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: ACM CCS 2017. pp. 2087–2104 (2017)
6. Asharov, G., Orlandi, C.: Calling out cheaters: Covert security with public verifiability. In: ASIACRYPT 2012. pp. 681–698. LNCS (2012)
7. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. In: TCC 2007. pp. 137–156. LNCS (2007)
8. Aviram, N., Gellert, K., Jager, T.: Session resumption protocols and efficient forward security for TLS 1.3 0-RTT. In: EUROCRYPT 2019. pp. 117–150. LNCS (2019)
9. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: EUROCRYPT 2012. pp. 263–280. LNCS (2012)
10. Bentov, I., Kumaresan, R., Miller, A.: Instantaneous decentralized poker. In: ASIACRYPT 2017. pp. 410–440. LNCS (2017)
11. Bernstein, D.J.: Chacha, a variant of salsa20. In: Workshop Record of SASC. vol. 8, pp. 3–5 (2008)
12. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: ASIACRYPT 2013. pp. 280–300. LNCS (2013)
13. Bowe, S.: BLS12-381: New zk-SNARK Elliptic Curve Construction (Mar 2017), `https://electriccoin.co/blog/new-snark-curve/`
14. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation. In: ACM CCS 2019. pp. 291–308 (2019)
15. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: PKC 2014. pp. 501–519. LNCS (2014)
16. Brakerski, Z., Vaikuntanathan, V.: Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In: TCC 2015. pp. 1–30. LNCS (2015)
17. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334 (2018)
18. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–88 (1981)
19. Chung, K.M., Lui, E., Pass, R.: From weak to strong zero-knowledge and applications. In: TCC 2015. pp. 66–92. LNCS (2015)
20. Dagher, G.G., Bünz, B., Bonneau, J., Clark, J., Boneh, D.: Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In: ACM CCS 2015. pp. 720–731 (2015)
21. Damgård, I., Orlandi, C., Simkin, M.: Black-box transformations from passive to covert security with public verifiability. In: CRYPTO 2020. pp. 647–676. LNCS (2020)
22. Fauzi, P., Meiklejohn, S., Mercer, R., Orlandi, C.: Quisquis: A new design for anonymous cryptocurrencies. In: ASIACRYPT 2019. pp. 649–678. LNCS (2019)
23. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO'86. pp. 186–194. LNCS (1987)
24. Fisher, R.A., Yates, F.: Statistical tables for biological, agricultural and medical research. Oliver and Boyd, Edinburgh, 3rd edn. (1949)
25. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: CRYPTO 2001. pp. 368–387. LNCS (2001)
26. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: 43rd ACM STOC. pp. 99–108 (2011)

27. Goldreich, O.: A uniform-complexity treatment of encryption and zero-knowledge. Journal of Cryptology **6**(1), 21–53 (1993)
28. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: 25th FOCS. pp. 464–479 (1984)
29. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. Journal of Cryptology **9**(3), 167–190 (Jun 1996)
30. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: PKC 2003. pp. 145–160. LNCS (2003)
31. Groth, J.: Linear algebra with sub-linear zero-knowledge arguments. In: CRYPTO 2009. pp. 192–208. LNCS (2009)
32. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016. pp. 305–326. LNCS (2016)
33. Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle. In: EUROCRYPT 2008. pp. 379–396. LNCS (2008)
34. Groth, J., Lu, S.: Verifiable shuffle of large size ciphertexts. In: PKC 2007. pp. 377–392. LNCS (2007)
35. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: 39th ACM STOC. pp. 21–30 (2007)
36. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: ACM CCS 2013. pp. 669–684 (2013)
37. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: 24th ACM STOC. pp. 723–732 (1992)
38. Kwon, A., Lazar, D., Devadas, S., Ford, B.: Riffle: An efficient communication system with strong anonymity. PoPETs **2016**(2), 115–134 (2016)
39. Lindell, Y.: How to simulate it - A tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046 (2016), `http://eprint.iacr.org/2016/046`
40. Micali, S.: CS proofs (extended abstracts). In: 35th FOCS. pp. 436–453 (1994)
41. Micciancio, D., Sorrell, J.: Simpler statistically sender private oblivious transfer from ideals of cyclotomic integers. In: ASIACRYPT 2020. pp. 381–407. LNCS (2020)
42. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: ACM CCS 2001. pp. 116–125 (2001)
43. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: CRYPTO'91. pp. 129–140. LNCS (1992)
44. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: CRYPTO 2008. pp. 554–571. LNCS (2008)
45. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In: EUROCRYPT'95. pp. 393–403. LNCS (1995)
46. Secure Hash Standard (SHS). National Institute of Standards and Technology, NIST FIPS PUB 180-4, U.S. Department of Commerce (Aug 2015)
47. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. National Institute of Standards and Technology, NIST FIPS PUB 202, U.S. Department of Commerce (Aug 2015)
48. de Valence, H., Grigg, J., Tankersley, G., Valsorda, F., Isis Lovecruft, M.H.: The ristretto255 and decaf448 groups. IETF CFRG Internet Draft (2020)