Efficient Succinct Zero-Knowledge Arguments in the CL Framework

Agathe Beaugrand^{1,2}, Guilhem Castagnos¹, and Fabien Laguillaumie²

¹ Université de Bordeaux, CNRS, INRIA, IMB, UMR 5251, F-33400 Talence, France. {agathe.beaugrand,guilhem.castagnos}@math.u-bordeaux.fr

² Université de Montpellier, CNRS, LIRMM, Montpellier, France. fabien.laguillaumie@lirmm.fr

Abstract. The CL cryptosystem, introduced by Castagnos and Laguillaumie in 2015, is a linearly homomorphic encryption scheme that has seen numerous developments and applications in recent years, particularly in the field of secure multiparty computation. Designing efficient zero-knowledge proofs for the CL framework is critical, especially for achieving adaptive security for such multiparty protocols. This is a challenging task due to the particularities of class groups of quadratic fields used to instantiate the groups of unknown order required in the CL framework.

In this work, we provide efficient proofs and arguments for statements involving a large number of ciphertexts. We propose a new batched proof for correctness of CL ciphertexts and new succinct arguments for correctness of a shuffle of these ciphertexts. Previous efficient proofs of shuffle for linearly homomorphic encryption were designed for Elgamal "in the exponent" which has only a limited homomorphic property.

In the line of a recent work by Braun, Damgård and Orlandi (CRYPTO 2023), all the new proofs and arguments provide *partial extractability*, a property that we formally introduce here. Thanks to this notion, we show that bulletproof techniques, which are in general implemented with groups of known prime order, can be applied in the CL framework despite the use of unknown order groups, giving non interactive arguments of logarithmic sizes.

To prove the practicability of our approach we have implemented these protocols with the BICYCL library, showing that computation and communication costs are competitive. We also illustrate that the partial extractability of our proofs provide enough guarantees for complex applications by presenting a bipartite private set intersection sum protocol which achieves security against malicious adversaries using CL encryption, removing limitations of a solution proposed by Miao *et al.* (CRYPTO 2020).

Keywords: class group cryptography, CL cryptosystem, linearly homomorphic encryption, ZK proofs, verifiable shuffle, multi-party computation

1 Introduction

The CL encryption scheme is a discrete-logarithm based linearly homomorphic encryption protocol, proposed by Castagnos and Laguillaumie in [CL15], and refined later in [CLT18]. It is built upon groups of unknown orders, and more particularly class groups of imaginary quadratic fields. Its security relies on a problem that is believed to be harder than the computation of discrete logarithms in multiplicative groups of finite fields or the factorization problem, which makes it more efficient in terms of communication and computational cost compared to other protocols, like Paillier encryption (*cf.* the implementation of [BCIL23]). It shares similarities with Elgamal "in the exponent", but suffers no limitation on the size of the message space, since the final discrete logarithm computation can be done in polynomial-time. An important consequence is that the message space can be as large as $\mathbf{Z}/q\mathbf{Z}$, where q is a prime number of several hundred bits.

When designing complex protocols with linearly homomorphic encryption, zero-knowledge (ZK) proofs are often required to ensure security against malicious adversaries by ensuring a certain behavior of the parties. ZK proofs and arguments related to CL encryption, such as proofs of correct generation of keys, proofs of correctness of ciphertexts or proofs of correct homomorphic operations have been first explored in [CCL⁺19,CCL⁺20]. In these works, two assumptions specific to class groups were used to ensure soundness of the proofs: the low order and strong root assumptions. Later, several works have improved efficiency or have proved soundness by relying on other assumptions [YCX21,DMZ⁺21]. More recently, [BDO23,BCD⁺24] use a new assumption called *rough order assumption* that greatly simplifies the design of ZK proofs for the CL framework. In [BDO23], a notion of *proof of plaintext knowledge* is also defined, which proves that a CL ciphertext actually encrypts a known plaintext, for which soundness is much easier to prove than in a proof of knowledge of the underlying ciphertext randomness.

ZK-argument of a shuffle. An important proof related to ciphertexts is a proof of a correct shuffle of ciphertexts. More precisely, a shuffle of ciphertexts (first exposed in [Cha81]) consists in mixing a set of ciphertexts c_1, \ldots, c_n to obtain a new set of re-randomized ciphertexts c'_1, \ldots, c'_n encrypting the same plaintexts but in a different order, such that the party that has encrypted the original ciphertexts can not link the encryptions. The main application of shuffles is mix-nets, that have real life applications such as e-voting [Gjø12,CEL⁺19] or anonymization of messages [Cha81,AKTZ17]. The shuffles we consider use semantically secure linear homomorphic encryption (as in [FS01,GI08]). The shuffler proceeds as follows: he chooses a random permutation π of $\{1, \ldots, n\}$ and randomness (ρ_i) and sets $c'_i = \text{Enc}(0; \rho_i) \otimes c_{\pi(i)}$, for $i = 1, \ldots n$, where \otimes denote the addition over encrypted data.

For most applications, in particular in the presence of malicious parties, the shuffling protocol must be *verifiable*. A formal definition of a verifiable shuffle is provided in [NSK04]. It requires to add to the shuffle a proof that the new

list of ciphertexts is indeed a shuffle of the old one, proof that must be at least complete and sound. Moreover, the notion of privacy states that an adversary cannot distinguish between two permutations given the transcript from a shuffle executed with one of them, even knowing the plaintexts and randomness in the first set of ciphertexts. This privacy is ensured by the zero-knowledge property of the proof. These ZK proofs represent the main efficiency bottleneck in protocols using verifiable shuffle, so a lot of work has been devoted to obtaining the best possible communication and computation costs.

Proofs of shuffle are mainly based on three approaches: the first one, from [Nef01], uses the invariance of a polynomial under the permutation of its roots. The second one uses the unique factorization of integers and was proposed by [Wik05]. Finally the third one uses a commitment to a permutation matrix, and a proof that the ciphertexts were shuffled according to this matrix, and was first proposed by [FS01]. In these three papers, the communication in the protocol is linear in the number of ciphertexts, and concerns the Elgamal encryption scheme. The permutation matrix approach was improved successively in [Fur04,GL07], and [GI08,BG12]. These last works present the first zero-knowledge arguments for a shuffle with sublinear communication for Elgamal encryption. Later, [HKR19] uses techniques coming from general arithmetic circuit satisfiability [BCC⁺16] and bulletproofs [BBB⁺18] to reduce the communication to logarithmic (at the expense of increasing the computation and the number of rounds). As for Paillier's cryptosystem, protocols for verifiable shuffles were proposed in [NSK04,NSK06], with a communication cost linear in the number of ciphertexts. Currently, no solution exists for CL encryption.

Application to PSI-sum Among the uses of shuffles, we can cite Private Set Intersection-sum (PSI-sum) or PSI-cardinality (PSI-CA) protocols. They are both variants of Private Set Intersection (PSI) protocols that allow two or more parties, each holding a private set of elements, to determine the intersection of their sets, without revealing any information about their own sets to the other parties. On the other hand, PSI-CA only outputs the cardinal of the intersection, without revealing its elements [HFH99]. In PSI-sum, introduced in [IKN⁺17], one of the parties associates a value to each element of its set, and the functionality outputs the sum of those values for the elements in the intersection, along with the cardinality, but without revealing anything else. A characteristic example of an application of PSI-sum is when a server knows statistics about a large population, and a user wants to compute an average on a subset of the population. This kind of scenarios can be found in different fields, from medical data to social networks or ads conversion rate. The PSI-sum functionality was investigated in [IKN+17,IKN+20] for an honest adversary, and in [MPR+20], for the first PSI-sum protocol secure against malicious adversaries. This protocol uses a shuffle of Elgamal ciphertexts that encrypts the values of the elements of the set, and the sum is recovered using the homomorphic properties of Elgamal "in the exponent", restricting the range of the values, due to a final discrete logarithm computation.

1.1 Our contributions

- We extend the notion of *Proof of Plaintext Knowledge* from [BDO23] to a notion of ZK proofs with *partial extractability*. Such proofs seem more natural in the context of CL encryption, and the protocols are more efficient. Moreover, we argue that these proofs, even with partial extractability, are sufficient for most applications. In this setting, we propose a batch proof for correctness of CL ciphertexts.
- We give the first verifiable shuffle protocol for CL encryption, with a sublinear communication and a variant with logarithmic communication using bulletproof techniques, techniques that were previously only used in the context of groups of prime order.
- We give an implementation of our new ZK-proofs and arguments that proves their efficiency.
- To illustrate the usefulness of our techniques, we present an application of the shuffle proof in a private intersection-sum protocol removing limitations of [MPR⁺20].

1.2 Technical overview

In the CL framework, the design of efficient zero-knowledge proofs raises some technical challenges, due to the unknown order of the underlying group. The first point worth considering is how to achieve soundness. In a classical sigma protocol with groups of prime order, soundness relies on the invertibility of x - x' modulo the group order, where x and x' are two different challenges. However, in a group with unknown composite order like class groups used in CL, the invertibility of x - x' is not guaranteed. This means that the classical techniques relying on special soundness do not apply, and we either have to use smaller spaces of challenges $[CCL^+19]$ – in which case we have to repeat the proof a few times to ensure a sufficient soundness level - or rely on one or more computational assumptions (e.g. $[CCL^+20]$). Thanks to the C-rough assumption, introduced in [BDO23] and precised in [BCD⁺24], one can use ZK proofs that are only proved to be sound in \mathcal{C} -rough class groups, even when statements are generated using the CL setup algorithm, that do not produce C-rough class groups in general. Working with a C-rough group and using challenges x, x' smaller than C ensures that x - x' is invertible modulo the group order and gives unconditionally a soundness of $1/\mathcal{C}$.

However, even in this particular case of C-rough groups, special soundness is still problematic to achieve, as a polynomial time extractor will not be able to compute a modular inverse even when it exists, given that the order cannot be computed efficiently.

Furthermore, this unknown and non prime order prevents the use of advanced techniques to get efficient proofs in the case of statements involving a large number n of CL ciphertexts: strategies to get compact shuffle proofs in the discrete logarithm setting with groups of prime order q often rely on linear algebra modulo q. For instance, the compact shuffle proof of [BG12] or the Bulletproof

techniques of [BBB⁺18,HKR19] use challenges of the form $(1, x, x^2, \ldots, x^{n-1})$ where x is sampled and sent by the verifier. This not only reduces the communication, but is also a cornerstone in showing extractability of witnesses. Indeed, this makes it possible to use a Vandermonde matrix whose invertibility modulo q is guaranteed when rewinding the proof with distinct x_i 's modulo q.

To overcome all these challenging obstacles, one needs to take a closer look at CL ciphertexts. One of the peculiarities of the CL scheme is that randomness and message do not live in the same subgroups. The CL framework uses a cyclic subgroup G with a decomposition $G \simeq F \times H$ where F has known prime order q and H has unknown (composite) order. A CL ciphertext is of the form $(h^{\rho}, \mathbf{pk}^{\rho} f^m)$, where $h, \mathbf{pk} \in H$ and $f \in F$. As a result, the randomness ρ must be treated as an integer but the plaintext m is defined modulo q and the decomposition in a direct product of subgroups makes it possible to treat ρ and m independently. As an integer, ρ prevents us from obtaining extraction using classical special soundness. Yet it only represents the randomness, which, semantically speaking, has little significance. Indeed, in most applications involving CL ciphertexts, the ZK proofs aim mainly to guarantee two things: first, that the ciphertexts are correctly constructed, and in particular can be decrypted; second, that the message obtained after decryption has an expected value. As a result when proving soundness in the case of a group with rough order, we first show that some exponent exists in H, and independently use techniques from the known prime order setting to show that some plaintext exists encoded in the group of known prime order F.

These observations led us to the definition of *partial extractability* of ZK proofs, generalizing the proof of plaintext knowledge for CL of [BDO23]. In a proof of correctness of ciphertexts, partial extractability translates to a proof ensuring that there exists some randomness ρ and a message m such that $c = \mathsf{Enc}_{\mathrm{CL}}(m; \rho)$, but only m is extractable from accepting transcripts. The same techniques can be broken down in a variety of proofs and arguments for different types of statements over CL ciphertexts. However, this does not solve all the problems. For instance, we still can not use a challenge set of the form $(1, x, x^2, \ldots, x^{n-1})$. When working over the integers, with large values of n the size of x^i explodes which is not only computationally prohibitive but also gives proofs with linear communications in n, as the prover sends a response involving an integer greater than x^{n-1} . Nevertheless, using challenges of the form (x_1,\ldots,x_n) , we show that we can still have efficient (partial) extraction by combining the use of the C-rough assumption and a tailored rewinding procedure. Note that using such challenges leads to a linear communication cost for the verifier. To solve this, we can consider that these vectors of challenges are generated using a PRNG for interactive proofs or consider non interactive versions, in which challenges are generated using hashing.

1.3 Related works and other applications of our verifiable shuffle

Recent works develop batched proofs for CL statements. In [BC24], Bartoli and Cascudo use linear secret sharing to build batched proofs that provide special

soundness as soon as the underlying secret sharing scheme has 2-reconstruction, without any computational assumption. However, contrary to our batch proof for correctness of CL ciphertexts, their communication cost is linear in the number n of statements. In [BCD⁺24], Braun *et al.* also recently propose a batched proof of correctness of CL ciphertexts. This proof uses challenges of the form $(1, x, x^2, \ldots, x^{n-1})$. As discussed above, this also leads to a communication linear in n.

Recently, [BGR⁺24] constructs a protocol for secure computation of differentially private sparse histograms that aggregates the inputs from a large number of clients. We note that their construction uses similar building blocks that we use for our application to private intersection-sum protocol: namely using CL encryption for oblivious evaluation of Dodis-Yampolskiy PRF, and a verifiable shuffle for CL ciphertexts. This shuffle is used in a black box way and [BGR⁺24] gives no protocol to instantiate it: only the works of [BG12] and [HKR19] that describes verifiable shuffles for Elgamal ciphertexts are cited. However, as we shall see in Section 4, adapting these techniques for CL ciphertexts is not straightforward. To sum up, [BGR⁺24] would benefit from our zero-knowledge arguments and demonstrates the usefulness of our verifiable shuffle. Our techniques could also be applied in the work from [SBDG23] that presents an e-voting protocol that needs a verifiable shuffle for CL ciphertexts.

1.4 Roadmap

In the following, after a section on preliminaries, we give in Section 3 a batching proof for correctness of CL ciphertexts. This proof is needed to be applied upstream of a shuffle argument for CL, and although relying of standard batching techniques for sigma protocols [GLSY04], its exposition is an opportunity to present in details our techniques sketched above. Section 4 is devoted to the shuffle argument. We use the same structure as the shuffle argument proposed by [GI08,BG12], composed of a product argument on commitments and an argument for a multiexponentiation of ciphertexts. Our techniques allow us to use a commitment scheme modulo q so we can use directly the same product argument than [GI08]. The core of our work is thus an argument of multiexponentiation for CL ciphertexts. We propose two different protocols for this proof. First, our techniques allow us to derive a sublinear argument for CL from [GI08]. Second, we adapt techniques from bulletproofs for the first time in a context of groups of unknown order, and expose a multiexponentiation argument with logarithmic communication. Then, we implement the proof for correctness of ciphertexts and both multiexponentiation arguments. We present in Section 5 timings and communication costs for statements up to 2^{18} ciphertexts, showing that our new protocols are practical. Finally, in Section 6, we briefly discuss the private set intersection-sum problem, and the advantages of using our shuffle argument for CL ciphertexts.

2 Preliminaries

2.1 General notations

Throughout the paper, bold lowercase letters will denote vectors while regular letters will denote a single integer or group element. Matrices will, in general, be denoted with bold capital letters.

Due to batching of proofs or encryptions in the protocol, the notation for a set of elements may vary according to the context. For example, a list $A = (a_i)_{1 \le i \le n}$ will be referred to as either a vector $\mathbf{a} = (a_i)_{1 \le i \le n}$ or a matrix $\mathbf{A} = (a_{i,j})_{\substack{1 \le i \le \ell \\ 1 \le j \le m}}$ with $a_{i,j} = (a_{(i-1)m+j})_{\substack{1 \le i \le \ell \\ 1 \le j \le m}}$ and $n = \ell m$. In this last case, for any $i \in \llbracket 1, \ell \rrbracket$,

 a_i denotes the *i*-th row of the matrix, *i.e.* a vector of length m.

In Subsection 4.4, we use a matricial notation for multiexponentiation over a group \mathbb{G} . Let M be a matrix of group elements $(g_{i,j})_{\substack{1 \leq i \leq m \\ 1 \leq j \leq N}} \in \mathbb{G}^{m \times N}$, *i.e.*,

Then for any $\boldsymbol{x} = (x_j)_{1 \le j \le N} \in \mathbf{Z}^N$, $\boldsymbol{M}.\boldsymbol{x} := \left[\prod_{j=1}^N g_{i,j}^{x_j}\right]_{1 \le i \le m} \in \mathbb{G}^m$. In this context, the external product of a matrix by a scalar is defined, for $\mu \in \mathbb{C}^m$.

In this context, the external product of a matrix by a scalar is defined, for $\mu \in \mathbf{Z}$, by $\mu \mathbf{M} := [g_{i,j}^{\mu}]_{\substack{1 \le i \le m \\ 1 \le j \le N}}$. Note that matrices and vectors whose coefficients are group elements are denoted with brackets, while integer matrices are delimited by parentheses.

To encapsulate CL statements and Pedersen commitments, we generalize these notations to pseudo-matrices whose elements lie in different groups on the different rows. Note that in this case, we still need the columns to all have the same structure. By an abuse of notation, we write $\mathcal{M}_{m \times N}$ as the set of such pseudo matrices, and \mathcal{V}_m the set of pseudo vectors of length m (*i.e.*, vectors whose components potentially lie in different groups).

2.2 CL encryption

We recall the CL encryption as it is presented in [BCIL23]. Let \widehat{G} be an abelian finite group with unknown order, that has a cyclic subgroup $F = \langle f \rangle$ of prime order q. The order of \widehat{G} is $q\widehat{s}$, where \widehat{s} is unknown and assumed to be prime to q. Let $h = x^q$ for some random element $x \in \widehat{G}$. We set g = fh, and let G be the cyclic subgroup of \widehat{G} generated by g. Then, if $H = \{x^q, x \in G\}$, we have by [BCIL23, Proposition 1], that $\langle h \rangle = H$ and $G \simeq F \times H$. Moreover, there exists an algorithm Solve_{DL} that computes discrete logarithms of elements of Fin basis f in polynomial time.

Let σ be a statistical distance parameter. We denote by \mathcal{D}_H a distribution over the integers such that the distribution $\{h^x, x \stackrel{\$}{\leftarrow} \mathcal{D}_H\}$ is at negligible distance – with respect to σ – from the uniform distribution in H. Assuming the knowledge of a bound $\tilde{s} > \hat{s}$, this distribution can be instantiated efficiently from a folded uniform distribution (see [Tuc20, Subsection 2.7.1]): in the following, \mathcal{D}_H will be the uniform distribution over $[0, 2^{\sigma-2}\tilde{s}]$, so that the statistical distance to the uniform is less than $2^{-\sigma}$.

This framework allows to construct an IND-CPA linearly homomorphic encryption scheme. It looks like an Elgamal "in the exponent", but with a much more efficient decryption procedure, which makes it possible to use a larger message space. We denote $\mathsf{Setup}_{\mathrm{CL}}(1^{\lambda}, q, \rho)$ the setup algorithm that generates such a group (the size of \hat{s} depends on λ), where ρ denotes the internal randomness of the algorithm. Its output is $pp_{\mathrm{CL}} := (\hat{G}, \tilde{s}, g, h, f, \mathsf{Solve}_{\mathrm{DL}})$, along with the information necessary to perform computations in G.

The CL encryption scheme is constructed as follows: a pair of keys is of the form (CL-sk, CL-pk) $\stackrel{\$}{\leftarrow}$ KeyGen_{CL}($\hat{G}, \tilde{s}, g, h, f$), with CL-sk $\stackrel{\$}{\leftarrow} \mathcal{D}_H$, and CL-pk $\leftarrow h^{\text{CL-sk}}$. A message $m \in \mathbb{Z}/q\mathbb{Z}$ is encrypted as a two-component ciphertext

$$c = \mathsf{Enc}_{\mathrm{CL}}(\mathsf{CL}\text{-}\mathsf{pk}; m; \rho) = (c_1, c_2) = (h^{\rho}, \mathsf{CL}\text{-}\mathsf{pk}^{\rho}f^m),$$

where $\rho \stackrel{\$}{\leftarrow} \mathcal{D}_H$. To decrypt such a ciphertext, one computes $d = c_2 \cdot c_1^{-\mathsf{CL-sk}} = f^m$ and runs the algorithm $\mathsf{Solve}_{\mathsf{DL}}$ on d to retrieve m. We denote $\mathsf{Dec}_{\mathsf{CL}}(\mathsf{CL-sk}; c)$ this operation.

One of the parameters (key or randomness) of Enc_{CL} or Dec_{CL} , when clearly defined, might be omitted to lighten notations.

Note that a ciphertext is an element of $H \times G$. However, the order of G being unknown, one can only check if an element is in \hat{G} or not, but cannot check its belonging to G (nor H). This remark must be kept in mind when designing proofs of knowledge for statements involving CL ciphertexts.

As already mentioned, the CL cryptosystem is linearly homomorphic. Let $c = (c_1, c_2)$ and $c' = (c'_1, c'_2)$ be CL ciphertexts encrypting respectively m and $m' \in \mathbb{Z}/q\mathbb{Z}$, then

$$c \otimes c' := (c_1 \cdot c'_1, c_2 \cdot c'_2)$$

is an encryption of m + m',

$$c^a := (c_1^a, c_2^a)$$

is an encryption of am for any integer $a \in \mathbb{Z}$.

Some useful variants of CL encryption. In the two-party context, we can define a variant of CL encryption with a shared key and two-party decryption. We use it in particular in Section 6. In that section we also use another variant of CL, vector CL, based on randomness reuse, that allows to encrypt many messages at the same time in a more efficient manner. We give more details about those variants in Appendix A.

2.3 Commitment scheme

Commitments are an essential building block in many multiparty computation protocols. In the CL framework, despite working with unknown order groups, the fact that the message space is $\mathbf{Z}/q\mathbf{Z}$ and the partial extractability property for ZK proof that we define, allows us to use commitments modulo q, instead of (less efficient) commitments over the integers. In the following, we use Pedersen commitments defined in a group (\mathcal{G}, \times) of prime order q (typically a subgroup of an elliptic curve). Let \mathfrak{g} and \mathfrak{h} be two generators such that the discrete logarithm of \mathfrak{g} in basis \mathfrak{h} is unknown. To commit to a value $a \in \mathbf{Z}/q\mathbf{Z}$, one chooses a random $r \in \mathbf{Z}/q\mathbf{Z}$, and computes

$$C = \mathfrak{g}^a \mathfrak{h}^r.$$

This commitment scheme is *perfectly hiding* (given C even a powerful adversary has no information on a), and *computationally binding* (any PPT algorithm has negligible probability of finding two different openings for a given C) under the discrete logarithm assumption in \mathcal{G} .

Pedersen commitments can be generalized to commit to m values in a unique commitment. Given m + 1 generators of \mathcal{G} , $(\mathfrak{g}_1, \ldots, \mathfrak{g}_m, \mathfrak{h})$, we denote

$$\mathsf{Com}(\boldsymbol{a};r) := \mathfrak{g}_1^{a_1} \dots \mathfrak{g}_m^{a_m} \mathfrak{h}^r,$$

the commitment to $\boldsymbol{a} = (a_1, \dots, a_m) \in (\mathbf{Z}/q\mathbf{Z})^m$ with randomness $r \in \mathbf{Z}/q\mathbf{Z}$.

With a same set of m + 1 keys, we can also commit to any vector of length k < m, by using the k + 1 first keys. We will denote $\mathsf{Setup}_{\mathrm{Ped}}$ a generator for these public parameters $pp_{\mathrm{Ped}} := (\mathcal{G}, q, \mathfrak{g}_1, \ldots, \mathfrak{g}_m, \mathfrak{h}), pp_{\mathrm{Ped}} \leftarrow \mathsf{Setup}_{\mathrm{Ped}}(1^{\lambda}, m).$

Finally, we can commit to a matrix M of size $\ell \times m$: for $r \in (\mathbb{Z}/q\mathbb{Z})^{\ell}$, Com(M; r) denotes the vector of length ℓ whose *i*-th component is a commitment to the *i*-th row of M using randomness r_i . The commitment scheme on vectors and matrices remains both perfectly hiding and biding under the discrete logarithm assumption.

2.4 Computational assumptions

In the class group setting, we use two computational assumptions: the first one is the HSM assumption ([CLT18]) that ensures that CL is IND-CPA; and the second one is the C-rough order assumption introduced in [BDO23], which basically states that given a randomly sampled class group, it is difficult to determine if its order is C-rough or not. We here recall this assumption, and explicit how we will use it in the following sections.

Assumption 1 (C-rough order assumption, [BDO23]) Let λ a security parameter, q a prime number, C > 0 an integer and A a PPT algorithm. We denote $\mathcal{D}_{CL}^{C,rough}$ the uniform distribution over the set

$$\left\{\rho \in \{0,1\}^{\lambda} \left| (\widehat{G},\widetilde{s},g,h,f) \leftarrow \mathsf{Setup}_{\mathrm{CL}}(1^{\lambda},q;\rho) \ \land \ \forall \, p < \mathcal{C} \, \textit{prime} \,, p \nmid \textit{ord}(\widehat{G}) \right\}.$$

We say that \mathcal{A} solves the \mathcal{C} -rough order (RO_C) problem if its advantage

$$\mathsf{Adv}_{\mathcal{A}}^{RO_{\mathcal{C}}}(\lambda) := \left| \mathbb{P}\left[1 \leftarrow \mathcal{A}(1^{\lambda}, \mathcal{U}(\{0, 1\}^{\lambda})) \right] - \mathbb{P}\left[1 \leftarrow \mathcal{A}(1^{\lambda}, \mathcal{D}_{\mathrm{CL}}^{\mathcal{C}, \mathsf{rough}}(\lambda, q))) \right] \right|$$

is non-negligible with respect to λ . We say that the C-rough assumption holds if no PPT algorithm solves the $RO_{\mathcal{C}}$ problem. **Definition 1.** Let λ be a security parameter and q a prime number. We denote by $\mathsf{Setup}_{\mathsf{CL}}^{\mathcal{C},\mathsf{rough}}(1^{\lambda},q)$ the output of the CL setup algorithm run with parameters $(1^{\lambda},q;\rho)$, where the internal randomness ρ follows $\mathcal{D}_{\mathsf{CL}}^{\mathcal{C},\mathsf{rough}}(\lambda,q)$.

In the following, we note that the soundness of proofs can be unconditionally proved whenever the order of the underlying group is C-rough. Unfortunately, such class groups are rare, and, due to the class number problem, there is no known way to generate, even *almost surely*, such a group. In particular, the above setup algorithm *cannot* be implemented practically. It is defined as an abstract object and will only be used as a tool to prove security. That is where the C-rough assumption intervenes. Roughly speaking, this assumption allows to show that a protocol that is secure against a PPT adversary – to be precised for each practical case – when the CL encryption is set up using $\mathsf{Setup}_{CL}^{C-\operatorname{rough}}$, is also secure when the group is generated using Setup_{CL} . In particular, our proofs, that are sound only in the C-rough order case, can still be used in protocols in which the CL parameters are generated by Setup_{CL} . Theorem 7 of $[\mathrm{BCD}^+24]$ gives a formalization of this fact:

Theorem 1 ([BCD⁺24]). Consider a PPT adversary \mathcal{A} against an efficient security game $\mathsf{Game}^{\mathcal{O}}_{\mathcal{A}}(\lambda)$ that generates CL public parameters via a single query to an oracle $pp_{\mathrm{CL}} \leftarrow \mathcal{O}$, and let $\mathsf{Game}^{\mathcal{O}}_{\mathcal{A}}(\lambda) = 1$ denote that \mathcal{A} wins the game. Define $\mathsf{Game}^{\mathsf{normal}}_{\mathcal{A}}(\lambda)$ and $\mathsf{Game}^{\mathsf{rough}}_{\mathcal{A}}(\lambda)$ to be $\mathsf{Game}^{\mathcal{O}}_{\mathcal{A}}(\lambda)$ respectively when \mathcal{O} samples $pp_{\mathrm{CL}} \leftarrow \mathsf{Setup}_{\mathrm{CL}}^{\mathcal{C}-\mathsf{rough}}(1^{\lambda}, q)$. Under the \mathcal{C} -rough assumption,

$$\mathbb{P}\left[\mathsf{Game}^{\mathsf{normal}}_{\mathcal{A}}(\lambda) = 1\right] \leq \mathbb{P}\left[\mathsf{Game}^{\mathsf{rough}}_{\mathcal{A}}(\lambda) = 1\right] + \mathsf{negl}(\lambda).$$

In particular, if no adversary wins $\mathsf{Game}_{\mathcal{A}}^{\mathsf{rough}}(\lambda) = 1$ with non-negligible probability, then the same holds for $\mathsf{Game}_{\mathcal{A}}^{\mathsf{normal}}(\lambda) = 1$

2.5 Zero-knowledge proofs and arguments

Let \mathcal{R} be a relation with domain $\mathcal{X} \times \mathcal{W}$. In an interactive proof or argument for \mathcal{R} on a statement x in \mathcal{X} , a prover, knowing a witness $w \in \mathcal{W}$ such that $\mathcal{R}(x, w)$, interacts with a verifier to convince him that the witness exists. In a proof of knowledge or argument of knowledge for \mathcal{R} on a statement x, the prover wants to convince the verifier that he knows the witness w. More formally, we use the following definitions.

Definition 2 (Honest Verifier Zero-Knowledge proofs and arguments).

Let λ be a security parameter, σ statistical parameter. An honest verifier zero-knowledge proof for a relation \mathcal{R} is an interactive protocol between a prover and a verifier that is

(i) Complete: if the prover really knows a witness and both parties act honestly, then the proof is accepted.

- (ii) Sound: a prover makes the verifier accept the proof for a false statement x (i.e., a statement x such that there exists no w such that $\mathcal{R}(x, w)$), only with negligible probability in λ .
- (iii) Honest Verifier ZK (HVZK): there exists a simulator, that, given a statement x and the verifier's challenges, produces a transcript of a proof such that the statistical distance between this simulated transcript and a real accepting transcript with a honest verifier is negligible in σ .

If the soundness is only computational, i.e., it holds for polynomial-time provers, the protocol is an HVZK argument for \mathcal{R} .

Definition 3 (HVZK proofs of knowledge, [DF02]). An honest verifier zero-knowledge proof of knowledge for a relation \mathcal{R} is an honest verifier zeroknowledge proof that moreover verifies knowledge soundness: there exist some functions of the security parameter, κ and ν , respectively the knowledge error and failure probability, and an algorithm \mathcal{E} that, given a statement x and oracle access to a prover P^* making the proof accepted with probability $\varepsilon \geq \kappa(\lambda)$ on x, outputs a witness w such that $\mathcal{R}(x, w)$ with probability $p \geq 1 - \nu(\lambda)$. Moreover, \mathcal{E} runs in expected time at most

$$\mathbb{E}[T_{\mathcal{E}^{P^*}}] \le \frac{poly(\lambda)}{\varepsilon - \kappa(\lambda)},$$

where the probability is taken over the random choices made by \mathcal{E} .

An argument of knowledge is defined in a similar way as an HVZK argument that also verifies knowledge soundness.

The 2-special soundness is a stronger notion (it implies knowledge soundness), defined as follows: given two accepting transcripts, for the same statement and prover's random coins, but different challenges, there exists an extractor \mathcal{E} that computes a witness for the statement x in polynomial time.

Aside from these classical definitions, we now define a new notion of *partial* extractability. The idea behind this is that for relations on CL ciphertexts, it is challenging to construct an efficient argument that guarantees the extractability of the randomness, whereas the plaintext can be extracted by usual techniques. This can be observed in particular in $[CCL^+19, CCL^+20]$. However, in many applications, we do not need to extract the randomness. Instead, we just need to check that the ciphertext is well-formed and that the plaintext it contains verifies a particular property. A partially extractable argument is sufficient to provide these guarantees.

Definition 4 (HVZK Proof with partial extractability). Let \mathcal{R} be a relation with witness domain $\mathcal{W} = \mathcal{W}_1 \times \mathcal{W}_2$. A \mathcal{W}_1 -extractable honest verifier zero-knowledge proof is an honest verifier zero-knowledge proof for which there exists an extracting algorithm \mathcal{E} as in Definition 3 except that, on input a statement x, \mathcal{E} outputs only a partial witness $w_1 \in \mathcal{W}_1$, i.e. w_1 such that there exists $w_2 \in \mathcal{W}_2$ such that $\mathcal{R}(x, (w_1, w_2))$ holds. An HVZK proof which is W_1 -extractable for the relation \mathcal{R} on the statement x will be written

$$HVZK-PwPE \{x; w_{ext} = w_1; w_2 \mid \mathcal{R}(x, (w_1, w_2))\}.$$

We define in a similar way a W_1 -extractable HVZK argument, and denote such arguments as HVZK-AwPE $\{x; w_{ext} = w_1; w_2 \mid \mathcal{R}(x, (w_1, w_2))\}$.

This definition extends the definition of a *Proof of Plaintext Knowledge* exposed in [BDO23], and can be seen as a particular case of *f*-extractability [BCKL08,GKP22], with *f* here being the projection $\mathcal{W}_1 \times \mathcal{W}_2 \to \mathcal{W}_1$.

3 Batching proofs of correctness of CL ciphertexts

A CL ciphertext is of the form $(h^{\rho}, \mathsf{CL-pk}^{\rho}f^m)$. It is in particular an element of $H \times G \subset \widehat{G} \times \widehat{G}$. However, due to the unknown order of \widehat{G} , one cannot check if the first component is indeed in H. Moreover, even when $c \in H \times G$, it is not guaranteed that it is a valid ciphertext, and that the recipient will be able to decrypt it. That is why protocols using CL ciphertexts need to ensure that a ciphertext can be decrypted through a proof of knowledge. Both [CCL⁺20,BDO23] have focused on designing proofs that CL ciphertexts are well formed. The former presents an argument of knowledge with special soundness, that relies on the strong root assumption and the low order assumption; while the later proposes a proof of plaintext knowledge, that also allows to extract the plaintext from two accepting transcripts.

Here, we expose an efficient proof that applies for n ciphertexts. It is, whenever \widehat{G} has \mathcal{C} -rough order, a $(\mathbb{Z}/q\mathbb{Z})^n$ -extractable HVZK proof for the relation such that for $(c_i)_{i \in [1,n]} \in (\widehat{G}^2)^n$, there exists $(\boldsymbol{m}, \boldsymbol{\rho}) \in (\mathbb{Z}/q\mathbb{Z})^n \times \mathbb{Z}^n$ and

$$\forall i \in \llbracket 1, n \rrbracket, c_i = \mathsf{Enc}_{\mathrm{CL}}(m_i; \rho_i).$$

The protocol is depicted in Fig. 1 and its security is proved in Theorem 2. We show that we can use the small exponents test of [BGR98] and adapt the techniques of [BDO23,BCD⁺24] to get partial extractability for the *n* plaintexts. As mentioned in the introduction, the soundness (and thus partial extractability) holds when the CL parameters are generated by $\mathsf{Setup}_{CL}^{\mathcal{C}-\mathsf{rough}}$, *i.e.* the base group has rough order. In practice, thanks to Theorem 1, this proof is still usable within a larger protocol in which the class group is generated by $\mathsf{Setup}_{CL}^{\mathcal{C}-\mathsf{rough}}$,

Theorem 2. Let λ a security parameter, σ a statistical parameter, q a prime number, an integer 0 < C < q, $(\widehat{G}, \widetilde{s}, g, h, f, \mathsf{Solve}_{\mathrm{DL}}) \leftarrow \mathsf{Setup}_{\mathrm{CL}}^{\mathcal{C}-\mathsf{rough}}(1^{\lambda}, q)$ the output of the rough CL setup algorithm and let $\mathsf{CL}\text{-pk} \in \widehat{G}$. Consider n elements $c_i \in \widehat{G}^2$, for $i \in [\![1, n]\!]$. The protocol of Figure 1 with challenge space $[\![0, \mathcal{C}[\![^n,$ and witness domain $(\mathbf{Z}/q\mathbf{Z})^n \times \mathbf{Z}^n$ is an HVZK proof $(\mathbf{Z}/q\mathbf{Z})^n$ -extractable for the relation

 $\mathsf{HVZK}\operatorname{-PwPE}\left\{(c_i)_{i\in\llbracket 1,n\rrbracket}; w_{ext} = \boldsymbol{m}; \boldsymbol{\rho} \mid \forall i \in \llbracket 1,n\rrbracket, c_i = (h^{\rho_i}, \mathsf{CL}\operatorname{-}\mathsf{pk}^{\rho_i}f^{m_i})\right\},$ where HVZK holds if $(\rho_i)_{i\in\llbracket 1,n\rrbracket} \in \llbracket 0, 2^{\sigma}\tilde{s}\rrbracket^n$. Statement: $pp_{CL} = (\widehat{G}, \widetilde{s}, g, h, f, \mathsf{Solve}_{DL}), \mathsf{CL-pk} \in \widehat{G}, (c_i)_{i \in [\![1,n]\!]} \in (\widehat{G}^2)^n$ Prover's input: $m \in (\mathbf{Z}/q\mathbf{Z})^n, (\rho_i)_{1 \le i \le n} \in \mathbf{Z}^n$

1. The prover samples $\widetilde{\rho} \stackrel{\$}{\leftarrow} \llbracket 0, 2^{2\sigma} n \, \mathcal{C} \widetilde{s} \rrbracket$ and $\widetilde{m} \stackrel{\$}{\leftarrow} \mathbf{Z}/q\mathbf{Z}$ and sends

 $\widetilde{c} = \mathsf{Enc}_{\mathrm{CL}}(\widetilde{m}; \widetilde{\rho}) = (h^{\widetilde{\rho}}, \mathsf{CL}\text{-pk}^{\widetilde{\rho}}f^{\widetilde{m}}).$

2. The verifier samples a random challenge $\boldsymbol{x} = (x_1, \ldots, x_n) \xleftarrow{\$} [\![0, \mathcal{C}]\!]^n$.

3. The prover computes and sends

$$\begin{cases} \widehat{\rho} = \widetilde{\rho} + \sum_{i=1}^{n} \rho_i x_i \in \mathbf{Z} \\ \widehat{m} = \widetilde{m} + \sum_{i=1}^{n} m_i x_i \in \mathbf{Z}/q\mathbf{Z} \end{cases}$$

4. The verifier accepts the proof if $(c_i)_{i \in [\![1,n]\!]} \in (\widehat{G}^2)^n$ and

$$\widetilde{c}\otimes \bigotimes_{i=1}^n c_i^{x_i} = \left(h^{\widehat{\rho}}, \mathsf{CL}\text{-}\mathsf{pk}^{\widehat{\rho}}f^{\widehat{m}}\right)$$

Fig. 1. HVZK-PwPE for correctness of CL ciphertexts

Proof. The protocol is complete by elementary inspection.

Soundness. Assume that there exists a prover P^* that makes the verifier accept with probability $4/\mathcal{C} + \mathsf{nonnegl}(\lambda)$. A standard rewinding procedure (described for completeness in Appendix B, Lemma 3) allows to find in expected polynomial time n pairs of accepting transcripts: for $i \in [\![1, n]\!]$,

$$(\widetilde{c}^{(i)}, \boldsymbol{x}^{(i,1)}, (\widehat{\rho}^{(i,1)}, \widehat{m}^{(i,1)}))$$
 and $(\widetilde{c}^{(i)}, \boldsymbol{x}^{(i,2)}, (\widehat{\rho}^{(i,2)}, \widehat{m}^{(i,2)}))$

such that the challenges $\boldsymbol{x}^{(i,1)}$ and $\boldsymbol{x}^{(i,2)}$ differ only by their *i*-th coordinate, *i.e.*, for all $j \neq i$, $x_j^{(i,1)} = x_j^{(i,2)}$ and $x_i^{(i,1)} \neq x_i^{(i,2)}$. For each $i \in [\![1,n]\!]$, by dividing the verification equations of the two transcripts we obtain,

$$c_i^{x_i^{(i,1)} - x_i^{(i,2)}} = \mathsf{Enc}_{\mathrm{CL}}(\widehat{m}^{(i,1)} - \widehat{m}^{(i,2)}; \widehat{\rho}^{(i,1)} - \widehat{\rho}^{(i,2)}).$$

The order $q\hat{s}$ of \hat{G} is \mathcal{C} -rough, as \hat{G} was generated by $\mathsf{Setup}_{\mathsf{CL}}^{\mathcal{C}-\mathsf{rough}}$. Then $x_i^{(i,1)} - x_i^{(i,2)}$ is invertible modulo $q\hat{s}$ as $0 < |x_i^{(i,1)} - x_i^{(i,2)}| < \mathcal{C}$. We denote $z_i \in \mathbf{Z}$ an inverse modulo $q\hat{s}$ and obtain

$$c_i = c_i^{z_i(x_i^{(i,1)} - x_i^{(i,2)})} = \mathsf{Enc}_{\mathrm{CL}}(z_i(\widehat{m}^{(i,1)} - \widehat{m}^{(i,2)}); z_i(\widehat{\rho}^{(i,1)} - \widehat{\rho}^{(i,2)})),$$

so that for any $i \in [1, n]$, the c_i 's are correct ciphertexts, which proves soundness.

Partial extractability. For the partial extractability, assume that there is a prover that makes the proof for a statement (c_i) accepting with probability $1/\mathcal{C} +$ nonnegl (λ) . Rewinding as before, we get *n* pairs of challenges $\boldsymbol{x}^{(i,1)}$ and $\boldsymbol{x}^{(i,2)}$ for accepting transcripts so that for every $i \in [\![1, n]\!]$,

$$c_i^{x_i^{(i,1)} - x_i^{(i,2)}} = \mathsf{Enc}_{\mathrm{CL}}(\widehat{m}^{(i,1)} - \widehat{m}^{(i,2)}; \widehat{\rho}^{(i,1)} - \widehat{\rho}^{(i,2)}).$$

Moreover, from soundness, with overwhelming probability, there exist $\boldsymbol{m} \in (\mathbf{Z}/q\mathbf{Z})^n$ and $\boldsymbol{\rho} \in \mathbf{Z}^n$ such that for $i \in [\![1,n]\!]$, $c_i = \mathsf{Enc}_{\mathrm{CL}}(m_i;\rho_i)$, so that

$$c_i^{x_i^{(i,1)} - x_i^{(i,2)}} = \mathsf{Enc}_{\mathrm{CL}}((x_i^{(i,1)} - x_i^{(i,2)})m_i; (x_i^{(i,1)} - x_i^{(i,2)})\rho_i).$$

Now, a same CL ciphertext can only encrypt one value so

$$(x_i^{(i,1)} - x_i^{(i,2)})m_i \equiv \widehat{m}^{(i,1)} - \widehat{m}^{(i,2)} \pmod{q},$$

Finally, we can extract for all $i \in [\![1, n]\!]$,

$$m_i \equiv (\widehat{m}^{(i,1)} - \widehat{m}^{(i,2)}) \cdot (x_i^{(i,1)} - x_i^{(i,2)})^{-1} \pmod{q}$$

Zero-knowledge. Finally, we show that the protocol is HVZK. Define $N = 2^{2\sigma} n \mathcal{C} \tilde{s}$. We define the following simulator: given a challenge $\boldsymbol{x} \in [\![0, \mathcal{C}]\!]^n$, the simulator samples uniformly $\rho \stackrel{\$}{\leftarrow} [\![0, N[\![, \hat{m} \stackrel{\$}{\leftarrow} \mathbf{Z}/q\mathbf{Z}, \text{ and computes}]$

$$\widetilde{c} = \left(h^{\widehat{\rho}}, \mathsf{CL}\text{-}\mathsf{pk}^{\widehat{\rho}}f^{\widehat{m}}\right) \otimes \bigotimes_{i=1}^n c_i^{-x_i}$$

The simulated transcript is $(\tilde{c}, \boldsymbol{x}, (\hat{\rho}, \hat{m}))$. We compute the statistical difference between the simulated and the real $\hat{\rho}$. Denote them respectively $\hat{\rho}_S$ and $\hat{\rho}_R$, and denote $R = \sum_{i=1}^n x_i \rho_i \in [0, n\mathcal{C}2^{\sigma}\tilde{s}]$. Then for any $a \in \mathbb{Z}$,

$$\mathbb{P}(\widehat{\rho}_S = a) = \begin{cases} \frac{1}{N} & \text{if } a \in [\![0, N[\![\\ 0 & \text{otherwise.} \end{cases}] \end{cases}$$

On the other hand,

$$\mathbb{P}(\widehat{\rho}_R = a) = \mathbb{P}(\widetilde{\rho}_R = a - R) = \begin{cases} \frac{1}{N} & \text{if } R \leq a \leq N + R - 1\\ 0 & \text{otherwise,} \end{cases}$$

so the statistical distance between $\hat{\rho}_R$ and $\hat{\rho}_S$ is

$$\frac{1}{2}\sum_{a=0}^{N+R-1} |\mathbb{P}(\hat{\rho}_R = a) - \mathbb{P}(\hat{\rho}_S = a)| = \frac{1}{2} \left(\sum_{a=0}^{R-1} \frac{1}{N} + \sum_{a=N}^{N+R-1} \frac{1}{N} \right) = \frac{R}{N} \le 2^{-\sigma},$$

and $\hat{\rho}_R$ and $\hat{\rho}_S$ are statistically indistinguishable.

Moreover, \hat{m} is uniformly distributed over $\mathbf{Z}/q\mathbf{Z}$ and independent of $\hat{\rho}$, both in the simulation and in an honest instantiation, so that the distribution of the simulated $(h^{\hat{\rho}}, \mathsf{CL-pk}^{\hat{\rho}}f^{\hat{m}})$ is indistinguishable from the real one.

Communication cost. In this protocol, the verifier sends n challenges, which makes the overall communication cost linear. However, one can use a pseudorandom generator \mathcal{P} (as suggested in [CPP17]). That way, the verifier only sends a seed x that allows the prover to generate $(x_1, \ldots, x_n) = \mathcal{P}(x)$. We can also place ourselves in the non interactive setting, in which challenges are generated by hashing the statement and the commitment. In both cases, we get a communication of $\mathcal{O}(\log(n))$ bits. In particular, the prover's response $\rho \in [\![0, (2^{\sigma} + 1)n2^{\sigma}\mathcal{C}\tilde{s}]\!]$ if $(\rho_i)_{i \in [\![1,n]\!]} \in [\![0, 2^{\sigma}\tilde{s}]\!]^n$. It is the first batched proof for CL ciphertexts with such logarithmic communication.

4 Shuffle argument of CL ciphertexts

We now focus on an argument for the shuffle relation. Concretely, the prover is supposed to have re-encrypted and mixed a set of ciphertexts with a random permutation, and wants to convince another party that he performed this operation correctly. The goal of the shuffle is to ensure a form of unlinkability between the original and the shuffled ciphertexts, so the prover does not want to reveal any information about the permutation or the re-randomization. Note that the shuffler usually did not encrypt the initial messages and so has no idea what is contained in the ciphertexts he shuffles.

Formally, we construct an HVZK argument with partial extractability that is \mathfrak{S}_n -extractable for the relation

$$\mathsf{HVZK}\mathsf{-}\mathsf{AwPE}\left\{\boldsymbol{c},\boldsymbol{c}'; w_{ext} = \pi; \boldsymbol{\rho} \,|\, \forall i \in \llbracket 1,n \rrbracket, c'_i = \mathsf{Enc}_{\mathrm{CL}}(0;\rho_i) \otimes c_{\pi(i)}\right\}$$

where \mathfrak{S}_n denotes the set of permutations of a set with *n* elements. Once again, both soundness and partial extractability is proved only when \widehat{G} has \mathcal{C} -rough order, but the proof can still be used when the class group is generated by $\mathsf{Setup}_{\mathrm{CL}}$ thanks to the \mathcal{C} -rough assumption.

4.1 Outline of the protocol

To construct our argument, we adapt to the CL framework the protocol presented in [BG12] for Elgamal. Let us give a simple intuition on how this protocol works: given a random vector \boldsymbol{x} , the shuffler proves, first that the weighted product $c = c^{\boldsymbol{x}}$ of original ciphertexts is actually equal to a re-randomization of the weighted product $(c')^{\boldsymbol{y}}$, where \boldsymbol{y} is known by the prover; and second, that the weights \boldsymbol{x} and \boldsymbol{y} are a permutation of one another. In particular, this means that there exists a randomness ρ and a permutation π such that

$$\bigotimes_{i=1}^{n} c_{i}^{x_{i}} = \mathsf{Enc}_{\mathrm{CL}}(0;\rho) \otimes \bigotimes_{i=1}^{n} (c_{i}')^{x_{\pi(i)}}$$

Finally, this equation implies, with overwhelming probability over the choice of x, that c' is a shuffle of c. Proving the first point corresponds to an argument

Statement: $pp_{\text{Ped}} = (\mathcal{G}, q, \mathfrak{g}_1, \dots, \mathfrak{g}_m, \mathfrak{h}), \ pp_{\text{CL}} = (\widehat{G}, \widetilde{s}, g, h, f, \text{Solve}_{\text{DL}}), \ \text{CL-pk} \in \widehat{G},$

 $c = (c_i)_{i \in [\![1,n]\!]}$ ciphertexts for CL-pk, $c' = (c'_i)_{i \in [\![1,n]\!]} \in (\widehat{G}^2)^n$ with $n = \ell m$ Prover's input: $\pi \in \mathfrak{S}_n, \ \boldsymbol{\rho} \in \mathbf{Z}^n$

1. The prover chooses $\mathbf{r}_T \xleftarrow{\$} (\mathbf{Z}/q\mathbf{Z})^{\ell}$, sets $T := (\pi^{-1}(i))_{i \in [\![1,n]\!]}$, computes and sends

$$\boldsymbol{C}_T := \operatorname{Com}(\boldsymbol{T}; \boldsymbol{r}_T) \in \mathcal{G}^c,$$

by committing T using Pedersen commitments for m values.

- 2. The verifier chooses and sends challenges $\boldsymbol{x} := (x_1, \ldots, x_n) \stackrel{\$}{\leftarrow} \llbracket 0, \mathcal{C} \rrbracket^n$.
- 3. The prover chooses $\mathbf{r}_X \xleftarrow{} (\mathbf{Z}/q\mathbf{Z})^{\ell}$, sets $X := (x_{\pi^{-1}(i)})_{i \in [\![1,n]\!]} \in \mathbf{Z}^n$, computes and sends

$$oldsymbol{C}_X := \operatorname{\mathsf{Com}}\left(oldsymbol{X};oldsymbol{r}_X
ight) \in \mathcal{G}^\ell$$

- 4. The verifier chooses and sends challenges $y, z \stackrel{\$}{\leftarrow} (\mathbf{Z}/q\mathbf{Z})^{\times}$.
- 5. Define Z := (z, z, ..., z). Let $C_{-Z} := \text{Com}(-Z; \mathbf{0})$ and $C_A := C_T^y \cdot C_X$. Compute $A := yT + X, B := A - Z, r_B := yr_T + r_X$ and $\theta := \sum_{i=1}^n \rho_i x_i \in \mathbf{Z}$. Prover and verifier compute $c := \bigotimes_{i=1}^n c_i^{(x_i)}$ and engage in:
 - a product argument to prove knowledge of ${m B}, {m r}_B$ such that

$$\boldsymbol{C}_A \cdot \boldsymbol{C}_{-Z} = \mathsf{Com}(\boldsymbol{B}; \boldsymbol{r}_B) \text{ and } \prod_{i=1}^n b_i = \prod_{i=1}^n (yi + x_i - z) \in \mathbf{Z}/q\mathbf{Z}$$

– a multiexponentiation argument proving partial knowledge of $\boldsymbol{X},\,\boldsymbol{r}_x$ and θ such that

$$c = \mathsf{Enc}_{\mathrm{CL}}(0; \theta) \otimes \bigotimes_{i=1}^{n} c_{i}^{x_{\pi^{-1}(i)}} \text{ and } C_{X} = \mathsf{Com}(X; r_{X}).$$

6. The verifier accepts if both arguments are valid.

Fig. 2. HVZK-AwPE for a shuffle of CL ciphertexts

of knowledge of the exponents in a multiexponentiation, while the second one is proved thanks to a product argument and the fact that the roots of a polynomial remain the same under permutation of the factors. The efficiency of this protocol lies in the efficiency of the two subprotocols, and more particularly of the multiexponentiation argument, that is the communication and computation bottlenecks in the context of CL ciphertexts.

Our protocol is depicted in Figure 2. It provides partial extractability: the permutation is extractable while the re-rerandomization is not. The main modification compared to the shuffle protocol of [BG12] is on the choice of the challenges in step 2 and on the definition of X. In this step, [BG12] uses challenges of the form (x, x^2, \ldots, x^n) modulo q, that allows to reduce the communication cost by just sending x. In our context, we need to define these elements over the integers as they are used as exponents of ciphertexts which belong to a group of unknown order. In order to keep a sublinear communication cost, we define the challenge as $(x_1, \ldots, x_n) \in \mathbf{Z}$, as in the proof of Fig. 1. Moreover, we define X to be the permutation of the x_i 's. To commit to X, *i.e.*, to commit to the permutation, one could think of using commitments over the integers. However, the CL framework, where the plaintext set is $\mathbf{Z}/q\mathbf{Z}$, and partial extractability allow to use a commitment scheme with domain $\mathbf{Z}/q\mathbf{Z}$ and to commit to X only modulo q. Like in shuffles for Elgamal, we can thus still use standard Pedersen commitments (for m values) that can be efficiently instantiated over an elliptic curve subgroup of prime order q. Moreover, as the product argument only involves commitments and no ciphertexts, using Pedersen commitments allows to reuse directly known product arguments. However, by modifying the form of the challenge, we cannot apply anymore extraction techniques based on Vandermonde matrices like in [BG12]. The situation is more complex than in the proof of Theorem 2 where we could use standard rewinding techniques for the n challenges, extracting independently the n plaintexts. Here, doing this to extract the permutation will result on extracting evaluations of different permutations for each of the rewindings. We thus need to rewind the n challenges for a common commitment. Fortunately, we prove in Appendix B, Lemma 4, that this can be done in expected polynomial time.

Another modification, with respect to [BG12], is that we swapped the roles of $\mathbf{c} = (c_i)_{i \in [\![1,n]\!]}$ and $\mathbf{c}' = (c'_i)_{i \in [\![1,n]\!]}$. This change, which may seem trivial, is linked to a technical issue that occurs in the proofs when using CL encryption compared to Elgamal. The role of those two families of ciphertexts are not equivalent for CL ciphertexts. We suppose indeed that the first set contains genuine ciphertexts, that have been proven to be correct when generated, using the protocol of Fig. 1. After the shuffle, the second one contains elements of $(\widehat{G})^2$, that may not be ciphertexts. Consequently, the shuffle proof in the CL framework has two distinct goals: it needs to convince the verifier that the two families of ciphertexts encrypt the same plaintexts, but it also has to ensure that the shuffled ciphertexts \mathbf{c}' are indeed ciphertexts.

We defer the proof of the overall shuffle argument to Theorem 4 in Subsection 4.3 as it depends on the extractability properties of the sub-protocols used. For the product argument, as already said, we can reuse without modification a known product argument over commitments. However, the multiexponentiation argument of ciphertexts requires particular attention in the CL setting and is the subject of the next subsection.

4.2 Sublinear argument for multiexponentiation of CL ciphertexts

Consider *n* CL ciphertexts $(c_i)_{1 \le i \le n}$, *n* integer exponents $(a_i)_{1 \le i \le n}$ smaller than q and Pedersen commitments over $\mathbf{Z}/q\mathbf{Z}$ of those integers. Our goal is to propose a ZK argument that proves that some ciphertext c is of the form

$$c = \mathsf{Enc}_{\mathrm{CL}}(0; \rho) \prod_{i=1}^{n} c_{i}^{a_{i}}.$$

Our protocol, depicted in Figure 3, follows the same general structure used in [GI08] for Elgamal. However, as we shall see, the soundness and the extractability of this protocol must be carefully analyzed in the CL setting.

Note that [BG12] gives a multiexponentiation argument for Elgamal that provides better communication costs than [GI08]. Unfortunately, this result does not transfer well to the CL framework. This protocol uses again in an essential manner challenges of the form (x, x^2, \ldots, x^n) modulo q, which results in a lot of challenging technical issues when switching to integer exponents for CL, but, unfortunately, in a linear communication cost for the responses of the prover.

As in the shuffle proof, we denote by ℓ and m two integers such that $n = \ell m$. The *n* ciphertexts and exponents are organized into two matrices

$$(c_{i,j})_{\substack{1 \le i \le \ell \\ 1 \le j \le m}}$$
 and $\mathbf{A} = (a_{i,j})_{\substack{1 \le i \le \ell \\ 1 \le j \le m}}$

The argument proves that $c = \text{Enc}_{CL}(0; \rho) \prod_{i=1}^{\ell} c_i^{a_i}$, where the vectors a_i are committed with ℓ Pedersen commitments for m values. The choice of m and ℓ is decisive for the communication and computation costs of the protocol. The communication cost in particular is $\mathcal{O}(\ell^2 + m)$. Choosing $m \sim n^{2/3}$ and $\ell \sim n^{1/3}$, we obtain a sublinear protocol, with a communication cost in $\mathcal{O}(n^{2/3})$ even if some elements sent by the prover are integers.

The overall idea of the protocol of [GI08] is as follows. If the prover sends all the $E_i := c_i^{a_i}$, then the verifier can check that the product has the expected value. However, he will not be able to check that the E_i were honestly computed (in particular computed with the same exponents as the ones contained in the commitment). Instead, the prover sends all the $E_{i,j} = c_j^{a_i}$, and the verifier can check that $c = \prod_{i=1}^{\ell} E_{i,i}$. The prover can now convince him that he computed honestly the $E_{i,j}$ by proving his knowledge of the exponents $\hat{a} = \sum_{i=1}^{\ell} x_i a_i$ in the ℓ equations $\prod_{i=1}^{\ell} E_{i,j}^{x_i} = c_j^{\hat{a}}$, where x_1, \ldots, x_{ℓ} are uniformly random challenges. With overwhelming probability over the choice of those challenges, the prover cannot convince the verifier without knowing A.

However, in the CL context, the ciphertexts are elements of a group of unknown order and exponents cannot be fully extracted. By using the partial extractability techniques exposed in Section 3, we can still prove that these exponents exist as integers and match some committed values modulo q. Moreover, it is possible to extract these values modulo q. More precisely, we prove in Theorem 3 that we can extract a matrix $\mathbf{A}' \in (\mathbf{Z}/q\mathbf{Z})^{\ell \times m}$ such that $\mathbf{A}' \equiv \mathbf{A} \pmod{q}$. This is sufficient for our purpose. Indeed, for CL ciphertexts, the values modulo q of the exponents encode all the information that affects the underlying plaintexts. We will thus be able to prove that c is a ciphertext that encrypts the same plaintext than $\prod_{i=1}^{\ell} c_i^{a_i}$, but c may have been computed using integer exponents that only match the a_i 's modulo q. This only affects the randomness of the ciphertext, that may not be known by the prover, which is not a concern for the application to the shuffle protocol.

Theorem 3. Let λ be a security parameter, σ a statistical parameter, $n = \ell m$, $pp_{\text{Ped}} \leftarrow \text{Setup}_{\text{Ped}}(\lambda, m)$ that defines a group \mathcal{G} of prime order q, an integer

 $\begin{array}{l} \textbf{Statement:} \ pp_{\mathrm{Ped}} = (\mathcal{G}, q, \mathfrak{g}_1, \dots, \mathfrak{g}_m, \mathfrak{h}), \ pp_{\mathrm{CL}} = (\widehat{G}, \widetilde{s}, g, h, f, \mathsf{Solve_{DL}}), \ \mathsf{CL-pk} \in \widehat{G}, \\ \widehat{G}, \\ \boldsymbol{c} = (c_i)_{i \in \llbracket 1, n \rrbracket} \ \text{ciphertexts for } \mathsf{CL-pk}, \ c \in \widehat{G}^2, \ \boldsymbol{C}_A \in \mathcal{G}^\ell \\ \textbf{Prover's input:} \ \rho \in \mathbf{Z}, \ \boldsymbol{A} \in \mathbf{Z}^{\ell \times m}, r_A \in (\mathbf{Z}/q\mathbf{Z})^\ell \end{array}$

1. The prover samples uniformly at random

 $\begin{cases} \boldsymbol{a}_{0} \stackrel{\$}{\leftarrow} \llbracket 0, 2^{\sigma} \ell \mathcal{C}^{2} \llbracket^{m} \\ r_{0} \stackrel{\$}{\leftarrow} \mathbf{Z}/q\mathbf{Z} \\ (b_{i,j})_{\substack{0 \le i \le \ell \\ 1 \le j \le \ell}} \stackrel{\$}{\leftarrow} (\mathbf{Z}/q\mathbf{Z})^{(\ell+1) \times \ell} \\ (s_{i,j})_{\substack{0 \le i \le \ell \\ 1 \le j \le \ell}} \stackrel{\$}{\leftarrow} [\mathbf{Z}/q\mathbf{Z})^{(\ell+1) \times \ell} \\ (\tau_{i,j})_{\substack{1 \le i \le \ell \\ 1 \le j \le \ell}} \stackrel{\$}{\leftarrow} \llbracket 0, 2^{\sigma} \tilde{s} \rrbracket^{\ell \times \ell} \\ (\tau_{0,j})_{1 \le j < \ell} \stackrel{\$}{\leftarrow} \llbracket 0, 2^{2\sigma} \ell \mathcal{C} \tilde{s} \rrbracket^{\ell} \\ \tau_{0,\ell} \stackrel{\$}{\leftarrow} \llbracket 0, \mathcal{B}' \rrbracket \end{cases}$

and resets $b_{\ell,\ell} \leftarrow -\sum_{i=1}^{\ell-1} b_{i,i}, s_{\ell,\ell} \leftarrow -\sum_{i=1}^{\ell-1} s_{i,i}, \tau_{\ell,\ell} \leftarrow \rho - \sum_{i=1}^{\ell-1} \tau_{i,i}.$

He computes and sends to the verifier the following

$$\begin{cases} C_{a_0} = \mathsf{Com}(\boldsymbol{a}_0; r_0) \\ \forall i \in \llbracket 0, \ell \rrbracket, j \in \llbracket 1, \ell \rrbracket, C_{b_{i,j}} = \mathsf{Com}(b_{i,j}; s_{i,j}) \\ \forall i \in \llbracket 0, \ell \rrbracket, j \in \llbracket 1, \ell \rrbracket, E_{i,j} = \mathsf{Enc}_{\mathrm{CL}}(b_{i,j}; \tau_{i,j}) \otimes \boldsymbol{c}_j^{\boldsymbol{a}_i} \end{cases}$$

- 2. The verifier checks that $E_{i,j} \in \widehat{G}^2$ for all i, j and samples $\boldsymbol{x} = (x_1, \ldots, x_\ell) \xleftarrow{\$} [\![0, \mathcal{C}[\!]^\ell, \text{ that he sends to the prover.}]$
- 3. The prover computes and sends

$$\begin{cases} \widehat{\boldsymbol{a}} = \boldsymbol{a}_0 + \sum_{i=1}^{\ell} x_i \boldsymbol{a}_i \in \mathbf{Z}^m \\ \widehat{r} = r_0 + \sum_{i=1}^{\ell} x_i r_{A,i} \in \mathbf{Z}/q\mathbf{Z} \\ \text{For } 1 \leq j \leq \ell, \ \widehat{b}_j = b_{0,j} + \sum_{i=1}^{\ell} x_i b_{i,j} \in \mathbf{Z}/q\mathbf{Z} \\ \text{For } 1 \leq j \leq \ell, \ \widehat{s}_j = s_{0,j} + \sum_{i=1}^{\ell} x_i s_{i,j} \in \mathbf{Z}/q\mathbf{Z} \\ \text{For } 1 \leq j \leq \ell, \ \widehat{\tau}_j = \tau_{0,j} + \sum_{i=1}^{\ell} x_i \tau_{i,j} \in \mathbf{Z}. \end{cases}$$

4. The verifier accepts the proof if the following equations hold:

$$\begin{cases} C_{a_0} \boldsymbol{C}_A^{\boldsymbol{x}} = \operatorname{Com}(\hat{\boldsymbol{a}}; \hat{r}); \\ \forall \ j \in \llbracket 1, \ell \rrbracket, C_{b_{0,j}} \prod_{i=1}^{\ell} C_{b_{i,j}}^{x_i} = \operatorname{Com}(\hat{b}_j; \hat{s}_j); \\ \forall \ j \in \llbracket 1, \ell \rrbracket, E_{0,j} \otimes \bigotimes_{i=1}^{\ell} E_{i,j}^{x_i} = \operatorname{Enc}_{\operatorname{CL}}(\hat{b}_j; \hat{\tau}_j) \otimes \boldsymbol{c}_j^{\hat{\boldsymbol{a}}}; \\ \prod_{i=1}^{\ell} C_{b_{i,i}} = \operatorname{Com}(0; 0); \prod_{i=1}^{\ell} E_{i,i} = c. \end{cases}$$

Fig. 3. Multiexponentiation HVZK-AwPE for CL encryption

 $0 < \mathcal{C} < q$, $(\widehat{G}, \widetilde{s}, g, h, f, \mathsf{Solve}_{\mathrm{DL}}) \leftarrow \mathsf{Setup}_{\mathrm{CL}}^{\mathcal{C}-\mathsf{rough}}(1^{\lambda}, q)$ and let $\mathcal{CL}\text{-pk} \in \widehat{G}$. Consider *n* ciphertexts $\mathbf{c} = (c_i)_{i \in [\![1,n]\!]}$ for $\mathcal{CL}\text{-pk}$, an element *c* of \widehat{G}^2 , and a vector of Pedersen commitments $\mathbf{C}_A \in \mathcal{G}^{\ell}$. The protocol exposed in Figure 3, with challenge space $[\![0,\mathcal{C}]\!]^{\ell}$ and witness domain $(\mathbf{Z}/q\mathbf{Z})^{\ell \times m} \times (\mathbf{Z}/q\mathbf{Z})^{\ell} \times \mathbf{Z} \times \mathbf{Z}^{\ell \times m}$, is an HVZK argument which is $(\mathbf{Z}/q\mathbf{Z})^{\ell \times m} \times (\mathbf{Z}/q\mathbf{Z})^{\ell}$ -extractable for the relation

$$\mathsf{HVZK}\text{-}\mathsf{AwPE}\left\{\begin{array}{ll} (c, \boldsymbol{c}, \boldsymbol{C}_{A}); \\ w_{ext} = (\boldsymbol{A}', \boldsymbol{r}_{A}); \\ (\rho, \boldsymbol{A}) \end{array} \middle| \begin{array}{l} \boldsymbol{C}_{A} = \mathsf{Com}(\boldsymbol{A}'; \boldsymbol{r}_{A}) \\ \wedge \boldsymbol{A} \equiv \boldsymbol{A}' \pmod{q} \\ \wedge c = \mathsf{Enc}_{\mathrm{CL}}(0; \rho) \otimes \boldsymbol{c}^{\boldsymbol{A}} \end{array} \right\},$$

where soundness holds on the discrete logarithm assumption in \mathcal{G} and HVZK holds if the coefficients of \mathbf{A} are in $[\![0, \mathcal{C}]\!]$, $\rho \in [\![0, \mathcal{B}[\![$ and $\mathcal{B}' = 2^{\sigma} \mathcal{C}(\mathcal{B} + (\ell - 1)2^{\sigma}\tilde{s})$.

Remark 1. We here allow the re-rerandomization to be in a larger space than the usual space of randomnesses as we will use the multiexponentiation argument in the shuffle with a re-rerandomization $\theta = \sum_{i=1}^{n} x_i \rho_i$, which gives $\mathcal{B} = n\mathcal{C}2^{\sigma}\tilde{s}$, leading to $\mathcal{B}' = 2^{2\sigma}\mathcal{C}\tilde{s}(n\mathcal{C} + (\ell - 1))$.

Proof. Completeness and zero-knowledge. The proof of completeness is elementary computations, and the zero-knowledge property can be proved as in the proof of Theorem 2.

Soundness. We assume that the prover has a probability $8/\mathcal{C} + \mathsf{nonnegl}(\lambda)$ to convince the verifier. As in the proof of Theorem 2, we rewind the argument until finding ℓ pairs of accepting transcripts with challenges $\boldsymbol{x}^{(k,1)}$ and $\boldsymbol{x}^{(k,2)}$ that differ only by their k-th coordinate. For $k \in [\![1,\ell]\!]$, $i \in [\![1,2]\!]$, we denote $\widehat{\boldsymbol{a}}^{(k,i)}, \widehat{\boldsymbol{r}}^{(k,i)}, \widehat{\boldsymbol{b}}^{(k,i)}, \widehat{\boldsymbol{s}}^{(k,i)}, \widehat{\boldsymbol{\tau}}^{(k,i)}$ the elements sent in the third step of the transcripts associated to challenge $\boldsymbol{x}^{(k,i)}$. For each of these transcripts, the verification equations give for every $k, j \in [\![1,\ell]\!]$,

$$\begin{cases} E_{0,j} \otimes \bigotimes_{\substack{i=1\\\ell}}^{\ell} E_{i,j}^{x_i^{(k,1)}} = \mathsf{Enc}_{\mathrm{CL}}(\widehat{b}_j^{(k,1)}; \widehat{\tau}_j^{(k,1)}) \otimes c_j^{\widehat{a}^{(k,1)}} \\ E_{0,j} \otimes \bigotimes_{i=1}^{\ell} E_{i,j}^{x_i^{(k,2)}} = \mathsf{Enc}_{\mathrm{CL}}(\widehat{b}_j^{(k,2)}; \widehat{\tau}_j^{(k,2)}) \otimes c_j^{\widehat{a}^{(k,2)}}. \end{cases}$$

Dividing the first row by the second one, and denoting $\delta_k := x_k^{(k,1)} - x_k^{(k,2)} \in \mathbf{Z}$, we obtain

$$E_{k,j}^{\delta_k} = \mathsf{Enc}_{\mathrm{CL}}(\widehat{b_j}^{(k,1)} - \widehat{b_j}^{(k,2)}; \widehat{\tau_j}^{(k,1)} - \widehat{\tau_j}^{(k,2)}) \otimes c_j^{\widehat{a}^{(k,1)} - \widehat{a}^{(k,2)}}.$$

Using that the order $q\hat{s}$ of the group \hat{G} is \mathcal{C} -rough, and that $|x_k^{(k,1)} - x_k^{(k,2)}| < \mathcal{C}$, δ_k is invertible modulo $q\hat{s}$. Let us denote $\gamma_k \in \mathbf{Z}$ an (unknown) inverse and define, as elements of \mathbf{Z} :

$$\begin{cases} b_{k,j} := \gamma_k \left(\widehat{b_j}^{(k,1)} - \widehat{b_j}^{(k,2)} \right) \\ \tau_{k,j} := \gamma_k \left(\widehat{\tau_j}^{(k,1)} - \widehat{\tau_j}^{(k,2)} \right) \\ a_k := \gamma_k \left(\widehat{a}^{(k,1)} - \widehat{a}^{(k,2)} \right), \end{cases}$$
(1)

which gives

$$E_{k,j} = \mathsf{Enc}_{\mathrm{CL}}(b_{k,j};\tau_{k,j}) \otimes \boldsymbol{c}_j^{\boldsymbol{a}_k}.$$
(2)

The same operations on the commitments equations give for $k, j \in [1, \ell]$,

$$\begin{cases} C_{A,k}^{\delta_k} = \mathsf{Com}(\widehat{a}^{(k,1)} - \widehat{a}^{(k,2)}; \widehat{r}^{(k,1)} - \widehat{r}^{(k,2)}) \\ C_{b_{k,j}}^{\delta_k} = \mathsf{Com}(\widehat{b_j}^{(k,1)} - \widehat{b_j}^{(k,2)}; \widehat{s_j}^{(k,1)} - \widehat{s_j}^{(k,2)}). \end{cases}$$

We define

$$\begin{cases} r_{A,k} := \gamma_k \left(\hat{r}^{(k,1)} - \hat{r}^{(k,2)} \right) \pmod{q} \\ s_{k,j} := \gamma_k \left(\hat{s_j}^{(k,1)} - \hat{s_j}^{(k,2)} \right) \pmod{q} \end{cases}$$
(3)

As $\delta_k \gamma_k = 1 \mod q$, (1) and (3) give

$$\begin{cases} C_{A,k} = C_{A,k}^{\gamma_k \delta_k} = \mathsf{Com}(\boldsymbol{a}_k; r_{A,k}) \\ C_{b_{k,j}} = \mathsf{Com}(b_{k,j}; s_{k,j}). \end{cases}$$

Denoting A the matrix with rows a_1, \ldots, a_ℓ , the first equation means that C_A is a commitment of the matrix $A' := A \pmod{q}$ with randomness r_A . Moreover, from the second equation, we have

$$\prod_{i=1}^{\ell} C_{b_{i,i}} = \mathsf{Com}\left(\sum_{i=1}^{\ell} b_{i,i}; \sum_{i=1}^{\ell} s_{i,i}\right) = \mathsf{Com}(0;0),$$

where the last equality holds from the verification round. By the binding property of the commitment, it means that $\sum_{i=1}^{\ell} b_{i,i} = \sum_{i=1}^{\ell} s_{i,i} = 0 \pmod{q}$. Finally, using the last check of the verification round and defining $\rho := \sum_{i=1}^{\ell} \tau_{i,i}$, we get from (2) and the fact that CL is homomorphic modulo q that

$$c = \bigotimes_{i=1}^{\ell} E_{i,i} = \mathsf{Enc}_{\mathrm{CL}}(0;\rho) \otimes \boldsymbol{c}^{\boldsymbol{A}},$$

which concludes the soundness.

Partial extractability. We have seen that \mathbf{A}' and \mathbf{r}_A are an opening of \mathbf{C}_A . Moreover there are extractable from (1) and (3) as the value of γ_k modulo q can be computed as an inverse of δ_k modulo q.

4.3 Proof of the sublinear shuffle argument

We now conclude on the security of the shuffle argument of Fig. 2 using the properties of our multiexponentiation argument. For the product argument, we used to protocol of [GI08] for compatibility of the parameters ℓ and m, s.t. $n = \ell m$.

Theorem 4. Let λ be a security parameter, σ a statistical parameter, $n = \ell m$, $pp_{\text{Ped}} \leftarrow \text{Setup}_{\text{Ped}}(\lambda, m)$ that defines a group \mathcal{G} of prime order q, an integer $0 < \mathcal{C} < q$, $(\widehat{G}, \widetilde{s}, g, h, f, \text{Solve}_{\text{DL}}) \leftarrow \text{Setup}_{\text{CL}}^{\mathcal{C}-\text{rough}}(1^{\lambda}, q)$ and let $\mathcal{CL}\text{-pk} \in \widehat{G}$. Consider n ciphertexts $\mathbf{c} = (c_i)_{1 \leq i \leq n}$ for $\mathcal{CL}\text{-pk}$, and n elements $\mathbf{c}' = (c'_i)_{1 \leq i \leq n}$ of \widehat{G}^2 . Using the product argument of [GI08] and the multiexponentiation argument of Fig. 3, the protocol exposed in Fig. 2, with challenge space $[\![0, \mathcal{C}[\![$ and witness domain $\mathfrak{S}_n \times \mathbf{Z}^n$, is an HVZK argument \mathfrak{S}_n -extractable for the relation

 $\mathsf{HVZK-AwPE}\left\{ \left. \boldsymbol{c}, \boldsymbol{c}'; w_{ext} = \pi; \boldsymbol{\rho} \right| \; \forall i \in [\![1, n]\!], c'_i = \mathsf{Enc}_{\mathrm{CL}}(0; \rho_i) \otimes c_{\pi(i)} \right\},\$

where soundness holds on the discrete logarithm assumption in \mathcal{G} and HVZK holds if $\forall i \in [\![1,n]\!], 0 \leq r_i \leq 2^{\sigma} \tilde{s}$.

Proof. Completeness and zero-knowledge. The completeness and zero-knowledge property are proved as in [BG12].

Soundness. Assume the prover makes the verifier accept with probability $8/\mathcal{C} + \mathsf{nonnegl}(\lambda)$. If the transcript is accepted for given challenges x, y, z, we rewind from the choice of z until we have n + 1 accepting transcripts for different challenges z_0, \ldots, z_n . We note $Z^{(j)} := (z_j, \ldots, z_j)$. We run the extractor on the product argument and obtain for $j \in [0, n]$ some $B^{(j)}, r_B^{(j)}$, defined modulo q, such that

$$\boldsymbol{C}_A \cdot \boldsymbol{C}_{-Z_j} = \mathsf{Com}(\boldsymbol{B}^{(j)}; \boldsymbol{r}_B^{(j)}) \text{ and } \prod_{i=1}^n b_i^{(j)} = \prod_{i=1}^n (yi + x_i - z_j) \in \mathbf{Z}/q\mathbf{Z}.$$

Let $A^{(j)} := B^{(j)} + Z^{(j)}$, then $C_A = \text{Com}(A^{(j)}; r_B^{(j)})$, and as the commitment is binding, $A, r_B := A^{(j)}, r_B^{(j)}$ is constant and does not depend on j. We thus have for all j,

$$\prod_{i=1}^{n} (a_i - z_j) = \prod_{i=1}^{n} (y_i + x_i - z_j) \in \mathbf{Z}/q\mathbf{Z}.$$

The two polynomials of degree n, $\prod_{i=1}^{n} (a_i - X)$ and $\prod_{i=1}^{n} (y_i + x_i - X)$, coincide on n + 1 values, so they are equal, and by unicity of the decomposition in a product of irreducible polynomials, there exists a permutation $\tau \in \mathfrak{S}_n$ such that for any $i \in [\![1, n]\!]$,

$$a_i = y\tau(i) + x_{\tau(i)} \in \mathbf{Z}/q\mathbf{Z}.$$

We denote U, r_U and V, r_V such that $C_X = \mathsf{Com}(U; r_U)$ and $C_T = \mathsf{Com}(V; r_V)$. All these quantities are independent of y, z, by the binding property, as C_X, C_T are sent before the choice of y, z. The equation $\mathsf{Com}(A; r_B) = C_A = C_T^y \cdot C_X$ implies, by the binding property, that A = yV + U and $r_B = yr_V + r_U$, which gives, for any $i \in [1, n]$,

$$a_i = yv_i + u_i = y\tau(i) + x_{\tau(i)} \in \mathbf{Z}/q\mathbf{Z}.$$

With overwhelming probability over the choice of y, this implies that

$$v_i \equiv \tau(i) \pmod{q}$$
 and $u_i \equiv x_{\tau(i)} \pmod{q}$. (4)

Denoting $T := (\tau(1), \tau(2), \ldots, \tau(n))$, this gives that $C_T = \mathsf{Com}(T; r_V)$.

Finally, we rewind on the value of \boldsymbol{x} , until we obtain 2n accepting transcripts for a same commitment \boldsymbol{C}_T , with challenges $\boldsymbol{x}^{(i,1)}$ and $\boldsymbol{x}^{(i,2)}$ that differ only by their *i*-th coordinate for all $i \in [\![1,n]\!]$. This specific rewinding for n coordinates fixing the challenge can be done in expected polynomial time as proven in Lemma 4. For each challenge, we repeat the same operations of rewinding over z, and we extract a permutation τ , which is committed to in \boldsymbol{C}_T , sent before \boldsymbol{x} , so that τ is independent of the challenge for \boldsymbol{x} . From the partial extractability of the multiexponentiation argument, we also know that for every i, there exist matrices of integers $\boldsymbol{U}^{(i,1)}, \boldsymbol{U}^{(i,2)}$ and integers $\rho^{(i,1)}, \rho^{(i,2)}$ such that

$$\begin{split} &\bigotimes_{j=1}^{n} \left(c_{j}'\right)^{x_{j}^{(i,1)}} = \mathrm{Enc}_{\mathrm{CL}}(0;\rho^{(i,1)}) \otimes \bigotimes_{j=1}^{n} c_{j}^{u_{j}^{(i,1)}} \\ &\bigotimes_{j=1}^{n} \left(c_{j}'\right)^{x_{j}^{(i,2)}} = \mathrm{Enc}_{\mathrm{CL}}(0;\rho^{(i,2)}) \otimes \bigotimes_{j=1}^{n} c_{j}^{u_{j}^{(i,2)}} \end{split}$$

which implies that

$$(c_i')^{x_i^{(i,1)} - x_i^{(i,2)}} = \mathsf{Enc}_{\mathrm{CL}}(0; \rho^{(i,1)} - \rho^{(i,2)}) \otimes \bigotimes_{j=1}^n c_j^{u_j^{(i,1)} - u_j^{(i,2)}}$$

The order $q\hat{s}$ is C-rough and $|x_i^{(i,1)} - x_i^{(i,2)}| < C$, so $x_i^{(i,1)} - x_i^{(i,2)}$ is invertible mod $q\hat{s}$. Denote w_i its inverse, we now have, for all $i \in [1, n]$,

$$c'_{i} = \mathsf{Enc}_{\mathrm{CL}}(0; w_{i}(\rho^{(i,1)} - \rho^{(i,2)})) \otimes \bigotimes_{j=1}^{n} c_{j}^{w_{i}(u_{j}^{(i,1)} - u_{j}^{(i,2)})}.$$
 (5)

The partial extractability of the multiexponentiation argument ensures moreover that $U^{(i,1)}$, $U^{(i,2)}$ match the values committed in $C_X^{(i,1)}$, $C_X^{(i,2)}$ modulo q. As a result, from (4), we know that for any $i, j \in [\![1, n]\!]$, and $k \in [\![1, 2]\!]$, $u_j^{(i,k)} \equiv x_{\tau(j)}^{(i,k)}$ (mod q). Using the fact that w_i is also the inverse of $x_i^{(i,1)} - x_i^{(i,2)}$ modulo q, and denoting $\pi := \tau^{-1}$, we get

$$w_i \left(u_{\pi(i)}^{(i,1)} - u_{\pi(i)}^{(i,2)} \right) \equiv 1 \pmod{q},$$

and for $j \neq \pi(i)$,

$$u_j^{(i,1)} - u_j^{(i,2)} \equiv 0 \pmod{q}.$$

Consequently, there exist integers $(\mu_{i,j})_{i,j\in[\![1,n]\!]}$ such that for all $i,j\in[\![1,n]\!]$,

$$w_i(u_j^{(i,1)} - u_j^{(i,2)}) = \delta_{j,\pi(i)} + \mu_{i,j}q,$$
(6)

where $\delta_{i,j}$ denotes the Kronecker delta. We know that the $(c_j)_{j \in [\![1,n]\!]}$ are ciphertexts, implying that for any $j \in [\![1,n]\!]$, c_j^q is an encryption of 0. In particular, there exists $s_j \in \mathbb{Z}$ such that $c_j^q = \mathsf{Enc}_{\mathrm{CL}}(0; s_j)$.

Combining this with (6) and injecting into (5), this implies that for any $i \in [\![1, n]\!]$,

$$\begin{aligned} c'_{i} &= \mathsf{Enc}_{\mathrm{CL}}(0; w_{i}(\rho^{(i,1)} - \rho^{(i,2)})) \otimes \bigotimes_{j=1}^{n} c_{j}^{\delta_{j,\pi(i)} + \mu_{i,j}q} \\ &= \mathsf{Enc}_{\mathrm{CL}}(0; w_{i}(\rho^{(i,1)} - \rho^{(i,2)})) \otimes \bigotimes_{j=1}^{n} \left(c_{j}^{q} \right)^{\mu_{i,j}} \otimes c_{\pi(i)} \\ &= \mathsf{Enc}_{\mathrm{CL}}(0; r_{i}) \otimes c_{\pi(i)} \end{aligned}$$

where $r_i = \sum_{i=1}^n \left(w_i(\rho_i - \rho'_i) + \sum_{j=1}^n \mu_{i,j} s_j \right)$, which concludes the proof of soundness.

Partial extractability. One can extract openings of C_A from the product argument and of C_X from the multiexponentiation argument. This allows to compute an opening of C_T , which from the soundness proof, gives the permutation $\tau = \pi^{-1}$.

Communication cost. We use the product argument of [GI08] and our multiexponentiation argument with $m \sim n^{2/3}$ and $\ell \sim n^{1/3}$. As a result, both protocols have communication cost in $\mathcal{O}(n^{2/3})$. The only step of our shuffle argument of Fig. 2 which is not sublinear in communication, is step 2 where the verifier sends the challenge \boldsymbol{x} . As for our batched proof for correctness of ciphertexts, we can consider a generation by a pseudo random generator or a non-interactive version, which give a total sublinear cost in $\mathcal{O}(n^{2/3})$. As already said, the techniques of [BG12] for the multiexponentiation can not be adapted to the CL framework to get a cost of $\mathcal{O}(n^{1/2})$.

To improve further the communication cost of the shuffle protocol, we thus consider in the next subsection a different multiexponentiation argument inspired by bulletproofs, using techniques introduced in $[BCC^{+}16, BBB^{+}18]$.

4.4 Logarithmic argument for multiexponentiation

In this subsection, we devise a shuffle argument for n CL ciphertexts with logarithmic communication. We still use the protocol of Fig. 2, but with parameters $\ell = 1$ and m = n, *i.e.*, we commit to n elements at once with one element of \mathcal{G} . In [HKR19, Appendix C], a shuffle argument for Elgamal ciphertexts with logarithmic communication is given, that also uses Bayer–Groth's blueprint of [BG12]. We reuse the product argument devised there, that also works for commitments defined in a cyclic group \mathcal{G} of prime order q. We thus focus on devising a multiexponentiation argument for CL ciphertexts with logarithmic communication. As bulletproofs do for arithmetic circuits and the multiexponentiation argument for Elgamal ciphertexts of [HKR19], we embed the multiexponentiation statement in a linear matrix preimage statement. These techniques were devised for prime

order groups, but we will prove that they can be adapted in the CL context thanks to partial extraction.

Using $\ell = 1$ and m = n, one wants to prove a statement of the form

$$c = \mathsf{Enc}_{\mathrm{CL}}(0; \rho) \otimes c^{a} \in \widehat{G}^{2} \text{ and } C_{a} = \mathsf{Com}(a; r_{a}) \in \mathcal{G},$$

where $\boldsymbol{a} \in \mathbf{Z}^n, \boldsymbol{r}_a \in \mathbf{Z}/q\mathbf{Z}$ and $\rho \in \mathbf{Z}$. Once again, we underline that semantically, only the value of \boldsymbol{a} modulo q really matters. Using the matricial notation for multiexponentiation defined in Subsection 2.1, the previous equation can be rewritten as

$$M. oldsymbol{\xi} = t$$

with
$$\boldsymbol{M} := egin{bmatrix} \mathfrak{g}_1 & \cdots & \mathfrak{g}_n & \mathfrak{h}_1 & 1_{\mathcal{G}} \\ c_1^{(1)} & \cdots & c_n^{(1)} & 1_{\mathcal{G}} & h \\ c_1^{(2)} & \cdots & c_n^{(2)} & 1_{\mathcal{G}} & \mathsf{CL-pk} \end{bmatrix}, \, \boldsymbol{\xi} := egin{bmatrix} \boldsymbol{a} \\ r_a \\ \rho \end{pmatrix}, \, ext{and} \, \boldsymbol{t} := egin{bmatrix} C_a \\ c_1^{(1)} \\ c_2^{(2)} \end{bmatrix}.$$

We stress that the first row of M and t are composed of elements of \mathcal{G} of order q, while the two others are composed of elements of \widehat{G} , of unknown order. To guarantee the soundness of the proof, we want to assume that the matrix M verifies the hard kernel assumption, *i.e.*, it is hard to find a non trivial kernel element. It is not a priori obvious that the assumption holds for the previous matrix, particularly as we have no control over the distribution of the second and third rows. To overcome this obstacle, as in [HKR19], we first add two supplementary commitment keys $\mathfrak{g}_{n+1}, \mathfrak{h}_2 \in \mathcal{G}$, and make the prover commit to ρ modulo q in $C_{\rho} = \mathfrak{g}_{n+1}^{\rho} \mathfrak{h}_2^{r_{\rho}}$ before starting the argument. Then we add two rows in M corresponding to C_{ρ} and $\mathbf{C}_{\rho} \cdot C_a$, giving

$$oldsymbol{M} := egin{bmatrix} \mathfrak{g}_1 \ \cdots \ \mathfrak{g}_n \ \mathfrak{g}_{n+1} \ \mathfrak{h}_1 \ \mathfrak{h}_2 \ \mathfrak{g}_1 \ \cdots \ \mathfrak{g}_n \ 1_G \ \mathfrak{h}_1 \ 1_G \ \mathfrak{h}_1 \ \mathfrak{h}_2 \ \mathfrak{g}_1 \ \cdots \ \mathfrak{g}_{n+1} \ \mathfrak{h}_2 \ \mathfrak{h}_2 \ \mathfrak{g}_{n+1} \ \mathfrak{h}_2 \ \mathfrak{h}_2 \ \mathfrak{g}_1 \ \cdots \ \mathfrak{g}_n \ \mathfrak{h}_1 \ \mathfrak{h}_2 \ \mathfrak{h}_2 \ \mathfrak{g}_1 \ \mathfrak{h}_2 \ \mathfrak$$

Now, M verifies the hard kernel assumption on the first row . Indeed, finding an element in the kernel of the first row is equivalent to finding an opening of Com(0;0), which is hard as the commitment scheme is binding.

It remains to explicit the argument of knowledge for the linear matrix preimage problem. It uses recursively techniques that are similar to those of [BG12] to shrink the statement and witness, until the proof is elementary. We denote Nthe number of columns of M and assume $N = k^d$, with k a small number. While d > 1, each step reduces the size of the statement by a factor k, and finally prove the shrunk statement once it reaches size k. In practice, we will choose for k a small power of 2 (k = 2, 4, 8). Note that keeping k small allows to use challenges of the form $(1, x, \ldots, x^{k-1})$ even when x is an integer and working with unknown order groups. Statement: $pp, M \in \mathcal{M}_{m \times N}$ and $t \in \mathcal{V}_N$ Prover's input: $\xi \in \mathbf{Z}^N$

- Base case: $\Sigma_{rec,1}, d = 1, N = k$
 - 1. The prover sends $\boldsymbol{\xi}$.
 - 2. The verifier accepts the proof if $M.\xi = t$.
- General case: $\Sigma_{rec,d}$, d > 1, $N = k^d$ 1. Split M in k submatrices of $\mathcal{M}_{m \times k^{d-1}}$, and $\boldsymbol{\xi}$ in k subvectors of $\mathcal{V}_{k^{d-1}}$:

For $1 \leq \ell \leq 2k - 1$, $\ell \neq k$, the prover computes and sends

$$oldsymbol{u}_\ell := \sum_{\substack{1 \leq i,j \leq k \ j-i = \ell-k}} oldsymbol{M}_j . oldsymbol{\xi}_i$$

- 2. The verifier picks a challenge $x \stackrel{\$}{\leftarrow} \llbracket 0, \mathcal{C} \llbracket$ and sends it to the prover. Set $\boldsymbol{x} := (1, x, \dots, x^{k-1}), \ \boldsymbol{y} := (x^{k-1}, \dots, x, 1)$ and $\boldsymbol{z} := (x^{2k-2}, \dots, x, 1)$.
- 3. Set $u_k := t$. Both parties compute

$$\widehat{\boldsymbol{M}} := \sum_{i=1}^k x_i \boldsymbol{M}_i ext{ and } \ \widehat{\boldsymbol{t}} = \sum_{\ell=1}^{2k-1} z_\ell \boldsymbol{u}_\ell$$

The prover computes $\hat{\boldsymbol{\xi}} = \sum_{i=1}^{k} y_i \boldsymbol{\xi}_i$, and both engage in an argument of knowledge $\Sigma_{rec,d-1}$ of $\hat{\boldsymbol{\xi}}$ for the shrunk relation $\widehat{\boldsymbol{M}}.\hat{\boldsymbol{\xi}} = \hat{\boldsymbol{t}}.$

Fig. 4. Recursive non ZK interactive argument Σ_{rec}

Similarly to [HKR19], we first expose in Fig. 4, a non ZK recursive version of the protocol. It works for any pseudo-matrix M and vector t such that M and t's coefficients are in either \mathcal{G} or \widehat{G} , and all columns have the same structure. We require moreover that the first row of M and t is composed of elements of \mathcal{G} and that M verifies the hard kernel assumption on the first row. We call such (M, t) a $(\widehat{G}, \mathcal{G})$ -admissible statement. Even though the protocol is valid for any class group \widehat{G} , soundness and partial extractability are only proved when \widehat{G} has \mathcal{C} -rough order, *e.g.* when it is sampled by $\mathsf{Setup}_{\mathsf{CL}}^{\mathcal{C}-\mathsf{rough}}$. More details on the use of such an argument are provided in Subsection 2.4.

We assume that the prover and the verifier know M and t and that the prover additionnally knows a witness $\boldsymbol{\xi} \in \mathbf{Z}^N$ such that $M.\boldsymbol{\xi} = t$.

When the size of the statement reaches k, the prover simply sends $\boldsymbol{\xi}$, and the verifier has to check if the equation holds. The following lemma adapts Lemma

3.10 of [HKR19] to the CL setting. In a nutshell, it allows, given a witness of size k^{d-1} from the corresponding step of the protocol, to go back up one step, and either prove that there exists a witness of size k^d , or break a computational assumption. Moreover, using the partial extractability technique, we prove that the value modulo q of the witness can be computed iteratively.

Lemma 1. Let λ be a security parameter, $pp_{\text{Ped}} \leftarrow \text{Setup}_{\text{Ped}}(\lambda, n+2)$ that defines a group \mathcal{G} of prime order q, an integer $0 < \mathcal{C} < q$, and let $pp_{\text{CL}} \leftarrow \text{Setup}_{\text{CL}}^{\mathcal{C}-\text{rough}}(1^{\lambda}, q)$. Assume we run a step of the recursive protocol of Fig. 4 on $a(\widehat{G}, \mathcal{G})$ -admissible statement \mathbf{M}, \mathbf{t} for $\Sigma_{rec.d}$, then the following properties hold:

1. Given 2k - 1 inputs $(\widehat{M}, \widehat{t})$ of $\Sigma_{rec,d-1}$, for which there exist witnesses $\widehat{\boldsymbol{\xi}} \in \mathbf{Z}^{k^{d-1}}$ s.t. for each instance, $\widehat{M}.\widehat{\boldsymbol{\xi}} = \widehat{\boldsymbol{t}}$, then there exists a witness $\boldsymbol{\xi}$ s.t. $\boldsymbol{M}.\boldsymbol{\xi} = \boldsymbol{t}$. Moreover, $\boldsymbol{\xi} \pmod{q}$ can be computed efficiently from $\widehat{\boldsymbol{\xi}} \pmod{q}$.

2. Given 2k inputs and witnesses for $\Sigma_{rec,d-1}$, either the witness $\boldsymbol{\xi}$ found in 1. for $\Sigma_{rec,d}$ is consistent, i.e., verifies $\boldsymbol{u}_{\ell} = \sum_{\substack{1 \leq i,j \leq k \\ j-i = \ell-k}} \boldsymbol{M}_j \boldsymbol{\xi}_i$ for all ℓ , or we find a non trivial element in the kernel of the first row of \boldsymbol{M} .

3. Given a non trivial element in the kernel of the first row of \widehat{M} , one can efficiently compute a non trivial element in the kernel of the first row of M.

Proof. We only detail the proof of point 1. Points 2 and 3 can be proved as in [HKR19] using the fact that the first row of the matrices is composed of elements of \mathcal{G} of known prime order q.

Let $(\widehat{\boldsymbol{M}}^{(i)}, \widehat{\boldsymbol{t}}^{(i)})$ be 2k-1 inputs of $\Sigma_{rec,d-1}$ along with witnesses $\widehat{\boldsymbol{\xi}}^{(i)}$. For all $i \in [\![1, 2k-1]\!]$, sets

so that $\widehat{\boldsymbol{t}}^{(i)} = \widehat{\boldsymbol{M}}^{(i)} \cdot \widehat{\boldsymbol{\xi}}^{(i)} = \left(\sum_{\ell=1}^{k} (x^{(i)})^{\ell-1} \boldsymbol{M}_{\ell}\right) \cdot \widehat{\boldsymbol{\xi}}^{(i)} = \left[\boldsymbol{M}_{1} \dots \boldsymbol{M}_{k}\right] \cdot \boldsymbol{w}^{(i)}, \ i.e.,$ $\boldsymbol{M} \cdot \boldsymbol{w}^{(i)} = \widehat{\boldsymbol{t}}^{(i)} = \sum_{\ell=1}^{2k-1} z_{\ell}^{(i)} \boldsymbol{u}_{\ell}.$ We denote $\boldsymbol{W} := (\boldsymbol{w}^{(1)} \dots \boldsymbol{w}^{(2k-1)}),$ and $\boldsymbol{X} := (\boldsymbol{z}^{(1)} \dots \boldsymbol{z}^{(2k-1)}).$ Due to the rough order of \widehat{G} , we know that the differences $x^{(i)} - x^{(j)}$ are invertible modulo $q\widehat{s}$, so that \boldsymbol{X} is an invertible (transposed) Vandermonde matrix modulo $q\widehat{s}.$ Let $\boldsymbol{V} = (v_{i,j})$ be a right inverse of \boldsymbol{X} modulo $q\widehat{s}.$ We have, for $i \in [\![1, 2k-1]\!],$

$$(\boldsymbol{M}.\boldsymbol{W}) \cdot \begin{pmatrix} v_{1,i} \\ \vdots \\ v_{2k-1,i} \end{pmatrix} = \left[\hat{\boldsymbol{t}}^{(1)} \dots \hat{\boldsymbol{t}}^{(2k-1)} \right] \cdot \begin{pmatrix} v_{1,i} \\ \vdots \\ v_{2k-1,i} \end{pmatrix} =$$
$$= \sum_{j=1}^{2k-1} v_{j,i} \hat{\boldsymbol{t}}^{(j)} = \sum_{1 \le \ell, j \le 2k-1} z_{\ell}^{(j)} v_{j,i} \boldsymbol{u}_{\ell} = \boldsymbol{u}_{i},$$

so $[\boldsymbol{u}_1 \dots \boldsymbol{u}_{2k-1}] = (\boldsymbol{M}.\boldsymbol{W}).\boldsymbol{V} = \boldsymbol{M}.(\boldsymbol{W}.\boldsymbol{V})$. In particular, taking for $\boldsymbol{\xi}$ the k-th column of $\boldsymbol{W}.\boldsymbol{V}$, we get $\boldsymbol{u}_k = \boldsymbol{t} = \boldsymbol{M}.\boldsymbol{\xi}$. Moreover, while we only know the existence of the inverse matrix \boldsymbol{V} modulo $q\hat{s}$ and cannot compute it, its value modulo q is efficiently computable from the values modulo q of $\hat{\boldsymbol{\xi}}^{(1)}, \dots, \hat{\boldsymbol{\xi}}^{(2k-1)}$.

We now add to this non ZK recursive argument the first step of a classical argument. The complete protocol is exposed in Figure 5. From Lemma 1, we can prove partial extractability and zero knowledge follows from the first step. This gives the following theorem.

Statement: $pp_{\text{Ped}} = (\mathcal{G}, q, \mathfrak{g}_1, \dots, \mathfrak{g}_{n+1}, \mathfrak{h}_1, \mathfrak{h}_2), \ pp_{\text{CL}} = (\widehat{G}, \widetilde{s}, g, h, f, \text{Solve}_{\text{DL}}),$ $\text{CL-pk} \in \widehat{G}, \ \boldsymbol{c} = (c_i)_{i \in [\![1,n]\!]} \text{ ciphertexts for CL-pk}, \ \boldsymbol{c} \in \widehat{G}^2, \ C_a \in \mathcal{G}$ **Prover's input:** $\rho \in \mathbf{Z}, \ \boldsymbol{a} \in \mathbf{Z}^n, r_a \in \mathbf{Z}/q\mathbf{Z}$

1. The prover samples $r_{\rho} \stackrel{\$}{\leftarrow} \mathbf{Z}/q\mathbf{Z}$, and $\mathbf{r} \stackrel{\$}{\leftarrow} \llbracket 0, 2^{\sigma} \mathcal{C}^{2} \llbracket^{n} \times \llbracket 0, 2^{\sigma} \mathcal{B} \mathcal{C} \llbracket \times \llbracket 0, 2^{\sigma} \mathcal{C} q \llbracket^{2}$. He compute and sends $C_{\rho} = \mathsf{Com}(\rho; r_{\rho})$ and $\mathbf{b} = \mathbf{M}.\mathbf{r}$, with

$$M = \begin{bmatrix} \mathfrak{g}_1 & \dots & \mathfrak{g}_n & \mathfrak{g}_{n+1} & \mathfrak{h}_1 & \mathfrak{h}_2 \\ \mathfrak{g}_1 & \dots & \mathfrak{g}_n & 1_G & \mathfrak{h}_1 & 1_G \\ 1_G & \dots & 1_G & \mathfrak{g}_{n+1} & 1_G & \mathfrak{h}_2 \\ c_1^{(1)} & \dots & c_n^{(1)} & h & 1_G & 1_G \\ c_1^{(2)} & \dots & c_n^{(2)} & \mathsf{CL-pk} & 1_G & 1_G \end{bmatrix}.$$

We denote $\boldsymbol{t} := \begin{bmatrix} C_{\rho} \cdot C_a \\ C_a \\ C_{\rho} \\ c \end{bmatrix}$ and $\boldsymbol{\xi} := \begin{pmatrix} \boldsymbol{a} \\ \rho \\ r_a \\ r_{\rho} \end{pmatrix}.$
2. The verifier chooses a challenge $e \stackrel{\$}{\leftarrow} \llbracket 0, \mathcal{C} \llbracket$ and sends it to the prover.

3. They both compute $\boldsymbol{u} := \boldsymbol{b} + e\boldsymbol{t}$ and engage in Σ_{rec} (Fig. 4) for proving knowledge of $\boldsymbol{\zeta} := \boldsymbol{r} + e\boldsymbol{\xi}$ such that $\boldsymbol{M}.\boldsymbol{\zeta} = \boldsymbol{u}$.

Fig. 5. Succinct multiexponentiation HVZK-AwPE for CL ciphertexts

Theorem 5. Let λ be a security parameter, σ a statistical parameter, $pp_{Ped} \leftarrow$ Setup_{Ped} $(\lambda, n+2)$ that defines a group \mathcal{G} of prime order q, an integer $0 < \mathcal{C} < q$, $(\widehat{G}, \widetilde{s}, g, h, f, Solve_{DL}) \leftarrow Setup_{CL}^{\mathcal{C}-rough}(1^{\lambda}, q)$ and let CL-pk $\in \widehat{G}$. Consider n ciphertexts $\mathbf{c} = (c_i)_{i \in [\![1,n]\!]}$ for CL-pk, an element c of \widehat{G}^2 , and a Pedersen commitment $\mathbf{C}_A \in \mathcal{G}$. The protocol exposed in Figure 5, with challenge space $[\![0,\mathcal{C}]\!]$ and witness domain $(\mathbf{Z}/q\mathbf{Z})^n \times \mathbf{Z}/q\mathbf{Z} \times \mathbf{Z} \times \mathbf{Z}^n$, is an HVZK argument which is $(\mathbf{Z}/q\mathbf{Z})^n \times \mathbf{Z}/q\mathbf{Z}$ -extractable for the relation

$$\mathsf{HVZK}\text{-}\mathsf{AwPE}\left\{\begin{array}{ll} (c, \boldsymbol{c}, \boldsymbol{C}_A); \\ w_{ext} = (\boldsymbol{a}', r_a); \\ (\rho, \boldsymbol{a}) \end{array} \middle| \begin{array}{l} C_a = \mathsf{Com}(\boldsymbol{a}'; r_a) \\ \wedge \ \boldsymbol{a} \equiv \boldsymbol{a}' \pmod{q} \\ \wedge \ c = \mathsf{Enc}_{\mathrm{CL}}(0; \rho) \otimes \boldsymbol{c}^{\boldsymbol{a}} \end{array} \right\},$$

where soundness holds on the discrete logarithm assumption in \mathcal{G} and HVZK holds if the coefficients of **a** are in $[0, \mathcal{C}]$, $\rho \in [0, \mathcal{B}]$.

Communication cost. The product argument of [HKR19] and our multiexponentiation argument of Fig. 5 have communication cost in $\mathcal{O}(\log_k(n))$. As before, we can obtain a shuffle argument with $\mathcal{O}(\log_k(n))$ communications, using both arguments with our shuffle argument of Fig. 2.

5 Implementation

To demonstrate the practicality of our methods, we have implemented NIZK variants (using the Fiat Shamir heuristic) of our new zero knowledge proofs and arguments, namely the proof for correctness of CL ciphertexts of Fig. 1, and the arguments for multiexponentiation for CL encryption of Fig. 3 and Fig. 5. Unless stated otherwise, all timings reported are performed on a M1 Macbook Pro using 8 cores.

At the lowest level, we use the BICYCL library [BCIL23] for the arithmetic of class groups, the RELIC library [AGM⁺] for the arithmetic of elliptic curves and the GMP library [GMP] for the arithmetic over \mathbf{Z} .

We use several optimizations to speed-up the implementation, notably the exponentiation and the multiexponentiation operations which are the most critical computations in our arguments. For the exponentiation in the elliptic curve, we use a wNAF method with a window size w = 5, and the core function of BICYCL for exponentiation in class groups. For multiexponentiations in both groups, we use the interleaving method with wNAFs of size w = 5 (cf. [HMV03, Algorithm 3.51]). For multiexponentiations involving a large number n of elements, we partition the computation in t multiexponentiations of n/t elements that are done in parallel on t cores.

For fixed base exponentiations, primarily in order to speed up the encryption of large sets of messages with CL, we make extensive use of pre-computations. More precisely, we use a variant of the interleaving method. Suppose we want to compute g^k for a fixed g and k of ℓ bits. For a parameter $v \ge 1$, let us denote $d = \lceil \ell/v \rceil$ and $g_j = g^{2^{jd}}$ for j = 0 to v - 1. Then, as sketched in p. 113 of [HMV03], one can use the interleaving method to compute g^k with a multiexponentiation with wNAFs involving the g_j 's and partial exponents k_j obtained from the decomposition in basis 2^d of k. As g is fixed, one can use a precomputation of the g_j^{2i+1} for i = 0 to $2^{w-2} - 1$. However, we have departed slightly from this approach, that uses a different wNAFs for each k_j 's. Instead, we compute a wNAF of k which is then partitioned in v sub-segments which are used to treat each g_j in the interleaving algorithm. This results experimentally in more sparse segments and speeds up the computation. Overall, this method uses d-1 squares and $\ell/(w+1)$ multiplications, and $v2^{w-2}$ precomputed elements.

	Statement		Proof			
n	Comp. (s)	Size (MB)	Size (kB)	Prover comp. (s)	Verifier comp. (s)	
2^{9}	1.4	1.7	0.634	0.011	0.092	
2^{12}	2.98	13.7	0.634	0.016	0.563	
2^{15}	14.95	109.7	0.635	0.049	4.469	
2^{18}	110.9	877.5	0.635	0.324	36.67	

Fig. 6. Timings and sizes for the HVZK-PwPE for correctness of n ciphertexts of Fig. 1

Using v = 87 and w = 11 (so that 44544 elements are precomputed), on a single core, this gives 1.33 ms to compute a fixed basis exponentiation whereas the BICYCL fixed basis function takes 6.29 ms (using only 4 precomputed elements). For instance, as shown in the next table, this enables us to perform 2^{18} CL encryptions in 109.7 s with a precomputation of 1.2 s.

We report the timings and sizes for the HVZK-PwPE for correctness of ciphertexts of Fig. 1 in Fig. 6. We use a security level of 128 bits which results in a 1827-bit fundamental discriminant for CL encryptions, and a 256-bit message space. For this argument and the next ones, we use statistical distance and soundness parameters equal to 2^{-128} .

We now report timings and sizes for the HVZK-AwPE for multiexponentiations of ciphertexts of Fig. 3 in Fig. 7. For this implementation and the next, we use Pedersen commitments on the NIST elliptic curve P256 and the CL message space matches the prime order q of the elliptic curve. In the first column, we give the decomposition of $n = m \cdot \ell$.

We finish by reporting timings and sizes for the succinct HVZK-AwPE for multiexponentiations of ciphertexts of Fig. 5 in Fig. 8. We use k = 2 for the recursive method of Fig. 4. This shows that for large size of n the bulletproof method applied to the CL setting becomes more interesting in terms of proof size, as expected, but also in terms of timings. In particular, for 2^{18} CL ciphertexts, the non-interactive HVZK argument takes only 0.01% of the data to be transmitted.

We do not report timings for the whole shuffle proof as we reuse product arguments from previous works. Moreover, these arguments use only compu-

	State	ement	Proof			
$n=m\cdot\ell$	Comp. (s)	Size (MB)	Size (MB)	Prover comp. (s)	Verifier comp. (s)	
$2^9 = 2^8 \cdot 2^1$	2.36	1.7	0.119	0.441	0.258	
$2^{12} = 2^{10} \cdot 2^2$	3.6	13.7	0.456	3.89	1.51	
$2^{15} = 2^{12} \cdot 2^3$	19.77	109.7	1.8	46.2	11.11	
$2^{18} = 2^{14} \cdot 2^4$	146.1	877.5	7.1	638.8	86.0	

Fig. 7. Timings and sizes for the HVZK-AwPE for multipexponentiations of n ciphertexts of Fig. 3

	State	ement	Proof			
n	Comp. (s)	Size (MB)	Size (kB)	Prover comp. (s)	Verifier comp. (s)	
2^{9}	1.53	1.7	54.7	2.26	0.626	
2^{12}	3.58	13.7	69.9	10.73	3.24	
2^{15}	21.91	109.7	85.1	76.21	24.39	
2^{18}	167.43	877.5	100.2	609.67	195.47	

Fig. 8. Timings and sizes for the succint HVZK-AwPE for multiexponentiations of n ciphertexts of Fig. 5

tations on Pedersen commitments and when implemented with elliptic curves, these computations have smaller cost compared to operations on CL ciphertexts. As a result, timings and communication costs of Figs. 7 and 8 give a rough idea of the total cost of the shuffle.

6 Shuffling CL ciphertexts in a PSI-sum protocol

We consider a private intersection-sum (PSI-sum) protocol between two parties A and B. Each party knows a secret set, respectively $X = (x_i)_{1 \le i \le n_A}$ and $Y = (y_j)_{1 \le j \le n_B}$, and A moreover holds a set V of integer values that are associated to each one of the elements of X. We set $I := \{i \in [\![1, n_A]\!], x_i \in X \cap Y\}$. The aim of the protocol is for both A and B to compute $v = \sum_{i \in I} v_i$ and |I|, without revealing anything else about the sets X and Y, nor about their intersection $X \cap Y$. In [MPR⁺20], Miao *et al.* give a two-sided protocol for PSI-sum that is secure against malicious adversaries, using a combination of Elgamal and Camenisch-Shoup encryption and the verifiable shuffle for Elgamal ciphertexts of [BG12] in order to apply an oblivious PRF (Dodis-Yampolskyi) on the elements of X and Y.

In this section, we give an application of our verifiable shuffle protocol for CL ciphertexts by devising a PSI-sum protocol using CL encryption. We omit the security proof of the protocol, as PSI is not the core of this paper. Nevertheless, we mention one detail about this proof, to explicit the way to use the C-rough assumption. All the ZK proofs and arguments developed in this paper being

sound only when the underlying class group has rough order, we first place ourselves in this case, in which we can use a security proof that is very similar to the one of $[MPR^+20]$. In this particular case, we thus show that there exists no PPT adversary able to distinguish a simulated protocol in the ideal world from the real protocol. In a second time, we consider the real protocol with a class group generated by Setup_{CL} (thus not necessarily rough), and Theorem 1 allows to prove that an adversary has no more advantage in distinguishing that it had in the rough case, which allows to conclude that the protocol is secure against malicious adversaries even in the non-rough case.

6.1 Sketch of the protocol

A high level description of the protocol for PSI-sum using CL encryption is outlined in Fig. 9. It follows the idea of [MPR⁺20], but we introduce the CL encryption in two places:

- We use CL encryption as the homomorphic encryption scheme used to encrypt the values of V, instead of Elgamal in the exponent. This change improves the capacity of the protocol, as it allows to use a larger message space, thanks to the efficient decryption process of CL ([MPR⁺20] needs a final discrete logarithm computation to retrieve v).
- We also use CL encryption instead of Camenisch-Shoup encryption in the computation of the distributed oblivious PRF (DOPRF). In this process, the encryption has to interact with commitments, and using CL encryption allows to use Pedersen commitments modulo a prime number q instead of integer commitments. Pedersen commitments are both lighter to compute, and easier to deal with in zero-knowledge arguments. We note that very recently [BGR⁺24] proposes a similar solution.

The protocol can be decomposed in five main steps:

- 1. Offline setup.
 - the class group, public parameters for CL encryption and the generators for the commitment scheme are generated transparently by one party;
 - a pair of vector CL keys $(\mathsf{CL-pk}_P, \mathsf{CL-sk}_P)$ (cf. App. A) and Elgamal keys are produced per party P;
 - a key for a two-party CL encryption is shared between the players (cf. Appendix A.1);
 - Each party proves with an argument of knowledge that CL keys are well-formed.
- 2. Commitment to the values. The parties commit to their inputs, and A encrypts the integer values. They prove they know openings of their commitments, and A proves correctness of the ciphertexts (using Fig. 1).
- 3. Distributed OPRF. The Dodis-Yampolskyi PRF ([DY05]) is defined as follows: given an element g of a group \mathcal{G} of order q, for a key $k \in \mathbb{Z}/q\mathbb{Z}$, and an element $x \in \mathbb{Z}/q\mathbb{Z}$, $F_k(x) := g^{1/k+x}$.



Fig. 9. Protocol for PSI-sum with CL encryption - Sketch

The parties compute the images of their sets by this PRF in a distributed and oblivious way: A holds a secret key k_A and a $x \in \mathbf{Z}/q\mathbf{Z}$, B holds a secret key k_B , and A and B compute in two party $F_{k_A+k_B}(x)$, without revealing information neither about their secret keys nor the value x. Only B obtains the output.

The computations require a linearly homomorphic encryption scheme to guarantee privacy, for which we use CL encryption. Moreover, as A and B have to compute a large number of PRF values, the computations are batched using vector CL encryption (cf. Appendix A.2). The process is explicited in Fig. 10.

We use ZK-arguments $\Sigma_1, \Sigma_2, \Sigma_3$ ensuring that each step is performed correctly:

- $-\Sigma_1$ is a HVZK-AoK of opening of the commitments C_a and C_{α} .
- Σ_2 is a HVZK-AwPE that the ciphertexts are correct, Bob knows the plaintext, and it is coherent with the content of C_{k_B} .
- Σ_3 is a HVZK-AwPE that Alice knows $a_{i,j}$, and $\alpha_{i,j}$, and performed the homomorphic operations correctly.
- 4. Shuffle and identification of the intersection. The DOPRF results and the encrypted values are shuffled. The parties identify the elements in the intersection and compute homomorphically an encryption of the sum. A proof of correctness for the shuffle is made with our protocol for CL and with [BG12] for Elgamal.
- 5. Two party decryption and output. After the previous steps, both parties obtain a ciphertext $c = (c_1, c_2)$ encrypted with a shared two party key such that tsk = tsk_A + tsk_B. Then A and B respectively compute $c'_2 = c_2 \cdot c_1^{-\text{tsk}_A}$ and $c''_2 = c_2 \cdot c_1^{-\text{tsk}_B}$, prove correct half decryption and exchange these values.

6.2 Switching from Elgamal in the exponent to CL

As already stated, the use of the CL encryption scheme in the PSI-sum protocol allows a larger range of messages, and makes the final decryption efficient.

In the protocol of [MPR⁺20], that uses Elgamal in the exponent, there must be some upper bound B on the values of V, in order that the sum v is not too large. We note that in their protocol after encrypting the v_i 's, A does not provide a proof that the resulting ciphertexts encrypts plaintexts $\langle B$. As a result, a malicious A could add a large integer to one of the values, which would prevent the other party to decrypt in the final step when this element is in the intersection. In this case, A can still remove this large integer and obtain the output. Although this does not constitute a proper attack as the malicious adversary can abort before the ideal functionality gives its output to the honest party, adding an argument that the ciphertexts are well formed can avoid this type of denial-of-service behaviour. Note that the proof of security provided in [MPR⁺20] did not take this into account. In particular, the simulator for malicious A is supposed to decrypt the ciphertexts of all the v_i to obtain the **Public parameters:** a public key for vector CL, CL - $\mathsf{pk} \in \widehat{G}^m$, a key tuple $\mathcal{K} \in \mathcal{G}^m$ for batched Pedersen commitments, a generator ${\mathcal P}$ for the PRF and commitments \boldsymbol{C}_x to \boldsymbol{M}_x , and C_{k_A}, C_{k_B} to k_A, k_B . Party AParty BInput $X = (x_i)_{1 \le i \le n} \in (\mathbf{Z}/q\mathbf{Z})^n$, written $M_x = (x_{i,j}) \in (\mathbf{Z}/q\mathbf{Z})^{\ell \times m}$, $k_A \in \mathbf{Z}/q\mathbf{Z}$ Input $k_B \in \mathbf{Z}/q\mathbf{Z}$ Secret key CL-sk associated to CL-pk A samples $M_a \xleftarrow{\ } (\mathbf{Z}/q\mathbf{Z})^{\ell \times m}$, $\boldsymbol{r}_a \stackrel{\$}{\leftarrow} (\mathbf{Z}/q\mathbf{Z})^{\ell}, \, \boldsymbol{r}_\alpha \stackrel{\$}{\leftarrow} (\mathbf{Z}/q\mathbf{Z})^{\ell}, \, \text{and computes}$ $lpha_{i,j} = a_{i,j}(k_A + x_{i,j}) ext{ and } \ C_a = ext{Com}(\boldsymbol{M}_a; \boldsymbol{r}_a), C_\alpha = ext{Com}(\boldsymbol{M}_\alpha; \boldsymbol{r}_\alpha)$ $\boldsymbol{C}_a, \boldsymbol{C}_\alpha, \boldsymbol{\Sigma}_1$ $\begin{vmatrix} B \text{ encrypts } k_B \mathbf{Id}_m \text{ in} \\ \mathbf{c}_{k_B} = \mathsf{Enc}_{\mathrm{CL}}(\mathsf{CL-pk}; k_B \mathbf{Id}_m; \mathbf{r}) \end{vmatrix}$ c_{k_B}, Σ_2 Let $M_{\beta} = k_B M_a + M_{\alpha} \in \mathbf{Z}/q\mathbf{Z}^{\ell \times m}$. A computes an encryption c_{β} of M_{β} by homomorphic operations: for $i \in [\![1, \ell]\!]$, $c_{\beta,i} = \bigotimes_{j=1}^{\cdots} \left(c_{k_B,j} \right)^{a_{i,j}} \otimes \mathsf{Enc}_{\mathrm{CL}}(\mathsf{CL-pk}; \boldsymbol{\alpha}_i; r_{\beta,i})$ She also computes $\mathcal{G}_{i,j} = a_{i,j}\mathcal{P}$. $\boldsymbol{\mathcal{G}}, \boldsymbol{c}_{\beta}, \boldsymbol{\Sigma}_{3}$ B decrypts c_{β} to obtain M_{β} , and computes $y_{i,j} := \beta_{i,j}^{-1} \mathcal{G}_{i,j}$. **Output of Bob:** $(y_{i,j})_{i \in \llbracket 1,\ell \rrbracket, j \in \llbracket 1,m \rrbracket}$ such that $y_{i,j} = F_{k_A+k_B}(x_{i,j})$.

Fig. 10. Protocol for batched computations of DOPRF images

input that it provides to the functionality. Then it must be able to abort at this point if one of the ciphertexts can not be decrypted. In particular, in order for this proof of security to hold, A needs to provide at least an argument that she knows the associated plaintexts.

Adding an argument that the ciphertexts are well-formed, *i.e.*, that they encrypt plaintexts $\langle B \rangle$, would require to use range proofs, that are expensive processes. For our solution with CL ciphertexts, although there is no bound on the plaintexts, an argument of well-formedness is needed in order to use our verifiable shuffle. Such an argument can be efficiently computed, as exposed in Section 3 and experimentally shown in Fig. 6.

Acknowledgements This research was supported by the French ANR Project ANR-21-CE39-0006 SANGRIA and the France 2030 ANR Project ANR-22-PECY-003 SecureCompute. The authors thank Lennart Braun for his feedback on the C-rough order assumption.

References

- AGM⁺. D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. RELIC is an Efficient LIbrary for Cryptography. https://github.com/ relic-toolkit/relic.
- AKTZ17. Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. MCMix: Anonymous messaging via secure multiparty computation. In Engin Kirda and Thomas Ristenpart, editors, USENIX Security 2017: 26th USENIX Security Symposium, pages 1217–1234. USENIX Association, August 2017.
- BBB⁺18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In 2018 IEEE Symposium on Security and Privacy, pages 315–334. IEEE Computer Society Press, May 2018.
- BC24. Claudia Bartoli and Ignacio Cascudo. On sigma-protocols and (packed) black-box secret sharing schemes. In Qiang Tang and Vanessa Teague, editors, PKC 2024: 27th International Conference on Theory and Practice of Public Key Cryptography, Part II, volume 14602 of Lecture Notes in Computer Science, pages 426–457. Springer, Cham, April 2024.
- BCC⁺16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, Advances in Cryptology – EUROCRYPT 2016, Part II, volume 9666 of Lecture Notes in Computer Science, pages 327–357. Springer, Berlin, Heidelberg, May 2016.
- BCD⁺24. Lennart Braun, Guilhem Castagnos, Ivan Damgård, Fabien Laguillaumie, Kelsey Melissaris, Claudio Orlandi, and Ida Tucker. An improved threshold homomorphic cryptosystem based on class groups. In Clemente Galdi and Duong Hieu Phan, editors, SCN 24: 14th International Conference on Security in Communication Networks, Part II, volume 14974 of Lecture Notes in Computer Science, pages 24–46. Springer, Cham, September 2024.

- BCIL23. Cyril Bouvier, Guilhem Castagnos, Laurent Imbert, and Fabien Laguillaumie. I want to ride my BICYCL : BICYCL implements CryptographY in CLass groups. Journal of Cryptology, 36(3):17, July 2023.
- BCKL08. Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, TCC 2008: 5th Theory of Cryptography Conference, volume 4948 of Lecture Notes in Computer Science, pages 356–374. Springer, Berlin, Heidelberg, March 2008.
- BDO23. Lennart Braun, Ivan Damgård, and Claudio Orlandi. Secure multiparty computation from threshold encryption based on class groups. In Helena Handschuh and Anna Lysyanskaya, editors, Advances in Cryptology - CRYPTO 2023, Part I, volume 14081 of Lecture Notes in Computer Science, pages 613–645. Springer, Cham, August 2023.
- BG12. Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, Advances in Cryptology – EUROCRYPT 2012, volume 7237 of Lecture Notes in Computer Science, pages 263–280. Springer, Berlin, Heidelberg, April 2012.
- BGR98. Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In Kaisa Nyberg, editor, Advances in Cryptology – EUROCRYPT'98, volume 1403 of Lecture Notes in Computer Science, pages 236–250. Springer, Berlin, Heidelberg, May / June 1998.
- BGR⁺24. Lennart Braun, Adrià Gascón, Mariana Raykova, Phillipp Schoppmann, and Karn Seth. Malicious security for sparse private histograms. Cryptology ePrint Archive, Report 2024/469, 2024.
- CCL⁺19. Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. In Alexandra Boldyreva and Daniele Micciancio, editors, Advances in Cryptology – CRYPTO 2019, Part III, volume 11694 of Lecture Notes in Computer Science, pages 191–221. Springer, Cham, August 2019.
- CCL⁺20. Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II, volume 12111 of Lecture Notes in Computer Science, pages 266–296. Springer, Cham, May 2020.
- CEL⁺19. Chris Culnane, Aleksander Essex, Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. Knights and knaves run elections: Internet voting and undetectable electoral fraud. *IEEE Security & Privacy*, 17(4):62–70, 2019.
- Cha81. David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.
- CL15. Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. In Kaisa Nyberg, editor, *Topics in Cryptology – CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 487– 505. Springer, Cham, April 2015.
- CLT18. Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In Thomas Peyrin and Steven Galbraith, editors, Advances in Cryptology – ASIACRYPT 2018, Part II, volume 11273 of Lecture Notes in Computer Science, pages 733–764. Springer, Cham, December 2018.

- CPP17. Geoffroy Couteau, Thomas Peters, and David Pointcheval. Removing the strong RSA assumption from arguments over the integers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, Advances in Cryptology – EURO-CRYPT 2017, Part II, volume 10211 of Lecture Notes in Computer Science, pages 321–350. Springer, Cham, April / May 2017.
- DF02. Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, Advances in Cryptology – ASIACRYPT 2002, volume 2501 of Lecture Notes in Computer Science, pages 125–142. Springer, Berlin, Heidelberg, December 2002.
- DMZ⁺21. Yi Deng, Shunli Ma, Xinxuan Zhang, Hailong Wang, Xuyang Song, and Xiang Xie. Promise Σ-protocol: How to construct efficient threshold ECDSA from encryptions based on class groups. In Mehdi Tibouchi and Huaxiong Wang, editors, Advances in Cryptology – ASIACRYPT 2021, Part IV, volume 13093 of Lecture Notes in Computer Science, pages 557–586. Springer, Cham, December 2021.
- DY05. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography, volume 3386 of Lecture Notes in Computer Science, pages 416–431. Springer, Berlin, Heidelberg, January 2005.
- FS01. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In Joe Kilian, editor, Advances in Cryptology – CRYPTO 2001, volume 2139 of Lecture Notes in Computer Science, pages 368–387. Springer, Berlin, Heidelberg, August 2001.
- Fur04. Jun Furukawa. Efficient, verifiable shuffle decryption and its requirement of unlinkability. In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 319–332. Springer, Berlin, Heidelberg, March 2004.
- GI08. Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Nigel P. Smart, editor, Advances in Cryptology – EUROCRYPT 2008, volume 4965 of Lecture Notes in Computer Science, pages 379–396. Springer, Berlin, Heidelberg, April 2008.
- Gjø12. Kristian Gjøsteen. The norwegian internet voting protocol. In Aggelos Kiayias and Helger Lipmaa, editors, *E-Voting and Identity*, pages 1–18, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- GKP22. Chaya Ganesh, Hamidreza Khoshakhlagh, and Roberto Parisella. NIWI and new notions of extraction for algebraic languages. In Clemente Galdi and Stanislaw Jarecki, editors, SCN 22: 13th International Conference on Security in Communication Networks, volume 13409 of Lecture Notes in Computer Science, pages 687–710. Springer, Cham, September 2022.
- GL07. Jens Groth and Steve Lu. Verifiable shuffle of large size ciphertexts. In Tatsuaki Okamoto and Xiaoyun Wang, editors, PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography, volume 4450 of Lecture Notes in Computer Science, pages 377–392. Springer, Berlin, Heidelberg, April 2007.
- GLSY04. Rosario Gennaro, Darren Leigh, R. Sundaram, and William S. Yerazunis. Batching Schnorr identification scheme with applications to privacypreserving authorization and low-bandwidth communication devices. In

Pil Joong Lee, editor, Advances in Cryptology – ASIACRYPT 2004, volume 3329 of Lecture Notes in Computer Science, pages 276–292. Springer, Berlin, Heidelberg, December 2004.

- GMP. GMP. The GNU Multiple Precision Arithmetic Library. https://gmplib. org/.
- HFH99. Bernardo A. Huberman, Matt Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, page 78–86, New York, NY, USA, 1999. Association for Computing Machinery.
- HKR19. Max Hoffmann, Michael Klooß, and Andy Rupp. Efficient zero-knowledge arguments in the discrete log setting, revisited. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, ACM CCS 2019: 26th Conference on Computer and Communications Security, pages 2093–2110. ACM Press, November 2019.
- HMV03. Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography.* Springer-Verlag, Berlin, Heidelberg, 2003.
- IKN⁺17. Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. Private intersection-sum protocol with applications to attributing aggregate ad conversions. Cryptology ePrint Archive, Report 2017/738, 2017.
- IKN⁺20. Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020, pages 370–389. IEEE, 2020.
- MPR⁺20. Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, and Moti Yung. Two-sided malicious security for private intersection-sum with cardinality. In Daniele Micciancio and Thomas Ristenpart, editors, Advances in Cryptology – CRYPTO 2020, Part III, volume 12172 of Lecture Notes in Computer Science, pages 3–33. Springer, Cham, August 2020.
- Nef01. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Michael K. Reiter and Pierangela Samarati, editors, ACM CCS 2001: 8th Conference on Computer and Communications Security, pages 116–125. ACM Press, November 2001.
- NSK04. Lan Nguyen, Reihaneh Safavi-Naini, and Kaoru Kurosawa. Verifiable shuffles: A formal model and a Paillier-based efficient construction with provable security. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, ACNS 04: 2nd International Conference on Applied Cryptography and Network Security, volume 3089 of Lecture Notes in Computer Science, pages 61-75. Springer, Berlin, Heidelberg, June 2004.
- NSK06. Lan Nguyen, Rei Safavi-Naini, and Kaoru Kurosawa. Verifiable shuffles: a formal model and a paillier-based three-round construction with provable security. *International Journal of Information Security*, 5:241–255, 2006.
- SBDG23. Chiara Spadafora, Michele Battagliola, Giuseppe D'Alconzo, and Andrea Gangemi. Multiparty class group encryption and applications to e-voting. CIFRIS23, 2023.
- Tuc20. Ida Tucker. Functional encryption and distributed signatures based on projective hash functions, the benefit of class groups. PhD thesis, École Normale Supérieure de Lyon, 2020.

- Wik05. Douglas Wikström. A sender verifiable mix-net and a new proof of a shuffle. In Bimal K. Roy, editor, Advances in Cryptology – ASIACRYPT 2005, volume 3788 of Lecture Notes in Computer Science, pages 273–292. Springer, Berlin, Heidelberg, December 2005.
- YCX21. Tsz Hon Yuen, Handong Cui, and Xiang Xie. Compact zero-knowledge proofs for threshold ECDSA with trustless setup. In Juan Garay, editor, PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part I, volume 12710 of Lecture Notes in Computer Science, pages 481–511. Springer, Cham, May 2021.

A Variants of CL encryption

A.1 Two-party CL encryption

Given a security parameter λ , a prime number q and $(\widehat{G}, \widetilde{s}, g, h, f, \mathsf{Solve}_{\mathsf{DL}})$, the output of $\mathsf{Setup}_{\mathsf{CL}}(1^{\lambda}, q)$, one can construct a two-party encryption scheme as follows:

- A generates its keys with $(\mathsf{CL}\text{-}\mathsf{sk}_A, \mathsf{CL}\text{-}\mathsf{pk}_A) \leftarrow \mathsf{KeyGen}_{\mathrm{CL}}(\widehat{G}, \widetilde{s}, g, h, f)$, similarly B generates $(\mathsf{CL}\text{-}\mathsf{sk}_B, \mathsf{CL}\text{-}\mathsf{pk}_B) \leftarrow \mathsf{KeyGen}_{\mathrm{CL}}(\widehat{G}, \widetilde{s}, g, h, f)$
- CL - $\mathsf{sk} \leftarrow \mathsf{CL}$ - $\mathsf{sk}_A + \mathsf{CL}$ - sk_B and CL - $\mathsf{pk} \leftarrow \mathsf{CL}$ - $\mathsf{pk}_A \cdot \mathsf{CL}$ - pk_B . The key CL - pk is public, while CL - sk is unknown to both A and B.
- The encryption is done normally using the public key CL-pk.
- The decryption is done in two party as follows: for a ciphertext $c = (c_1, c_2)$, A computes $c'_2 = c_2 \cdot c_1^{-\mathsf{CL-sk}_A}$, and sends it to B. Then B can decrypt normally, computing first $d = c'_2 \cdot c_1^{-\mathsf{CL-sk}_B}$, and retrieving the message by m =Solve_{DL}(d). If both parties want to retrieve the message, these operations are done symetrically.

In the case in which A can be malicious, B can ask him to prove in a zeroknowledge argument (or HVZK-AwPE) that he performed the half decryption honestly.

We use this encryption scheme in the PSI protocol exposed in Figure 9 to encrypt the values, so that both parties can decrypt together the sum in the last step.

A.2 Vector CL encryption

For efficiency purpose, we can reduce the size of CL ciphertexts for a batch of m messages by sharing randomness. Instead of using one randomness for each message to be encrypted, we only use one for the vector. To allow this, each key (public or private) will be a vector of m keys, *i.e.*, if $\mathbf{sk} = (\mathsf{CL-sk}_1, \ldots, \mathsf{CL-sk}_m)$, then the associated public key is $\mathbf{pk} = (\mathsf{CL-pk}_1, \ldots, \mathsf{CL-pk}_m) = (h^{\mathsf{CL-sk}_1}, \ldots, h^{\mathsf{CL-sk}_m})$. For $\boldsymbol{a} = (a_1, \ldots, a_m)$ a vector in $(\mathbf{Z}/q\mathbf{Z})^m$, we denote $\mathsf{Enc}_{\mathrm{CL}}(\mathbf{pk}; \boldsymbol{a}; r)$ the

encryption of \boldsymbol{a} under the public key \mathbf{pk} , and with randomness $r \stackrel{\$}{\leftarrow} \mathcal{D}_H$. It stands for

$$\mathsf{Enc}_{\mathrm{CL}}(\mathbf{pk}; \boldsymbol{a}; r) = (h^r, \mathsf{CL}\text{-}\mathsf{pk}_1^r f^{a_1}, \dots, \mathsf{CL}\text{-}\mathsf{pk}_m^r f^{a_m}).$$

With this transformation, a ciphertext encrypts m values in m + 1 components instead of 2m if we encrypted separately the values.

We denote $\mathsf{Dec}_{CL}(\mathbf{sk}; c)$ the decryption of c under the secret key \mathbf{sk} , which is computed as follows:

$$\mathsf{Dec}_{\mathrm{CL}}(\mathbf{sk}; c_0, \dots, c_m) = \left(\mathsf{Solve}_{\mathrm{DL}}\left(c_1 \cdot c_0^{-\mathsf{CL}\mathsf{-}\mathsf{sk}_1}\right), \dots, \mathsf{Solve}_{\mathrm{DL}}\left(c_m \cdot c_0^{-\mathsf{CL}\mathsf{-}\mathsf{sk}_m}\right)\right).$$

In order to simplify the notations, we also define the encryption of a matrix $M_a = (a_i)_{i \in [\![1,\ell]\!]}$ of size $\ell \times m$. If **pk** is a public key and $\mathbf{r} = (r_1, \ldots, r_\ell)$ is a vector of ℓ randomnesses,

$$\mathsf{Enc}_{\mathrm{CL}}(\mathbf{pk}; \boldsymbol{M}_a; \boldsymbol{r})$$

denotes a vector of ℓ ciphertexts, such that for every $i \in [\![1, \ell]\!]$, the *i*-th ciphertext encrypts the *i*-th row of M_a with randomness r_i , ie

$$\mathsf{Enc}_{\mathrm{CL}}(\mathbf{pk}; \boldsymbol{M}_a; \boldsymbol{r}) = (\boldsymbol{c}_1, \dots, \boldsymbol{c}_\ell),$$

with for every $1 \leq i \leq \ell$,

$$c_i = \mathsf{Enc}_{\mathrm{CL}}(\mathbf{pk}; a_i; r_i).$$

The encryption of a matrix of size $\ell \times m$ thus contains $\ell(m+1)$ elements of the class group.

B Rewinding procedures

We consider a ZK protocol in which the challenge sent by the verifier is of the form $\boldsymbol{x} = (x_1, \ldots, x_n)$ where the x_i 's are independent and uniformly random in $[0, \mathcal{C}[$ and n > 1. Figure 1 is an example of such a protocol. We fix a statement \mathcal{S} and a prover \mathcal{P} that has probability ε of making the verifier accept the proof on the statement \mathcal{S} .

We design an algorithm that outputs in expected polynomial time a set of challenges $\boldsymbol{x}_1, \boldsymbol{x}'_1, \ldots, \boldsymbol{x}_n, \boldsymbol{x}'_n$ such that for any $i \in [\![1, n]\!]$, the coordinates of \boldsymbol{x}_i and \boldsymbol{x}'_i are the same, except for the *i*-th coordinate. Moreover, for all *i*, there exists a set of random coins \mathcal{R}_i such that, if the prover uses \mathcal{R}_i as its random coins, then the proof is accepting for both challenges \boldsymbol{x}_i and \boldsymbol{x}'_i .

The next two lemmas essentially follows the rewinding algorithm of [DF02], slightly adapted to match the case in which the challenge has n independent components. They are exposed mainly for completeness, and to allow a better understanding of Lemma 4.

To visualize the situation, we define M the success matrix of \mathcal{P} on the statement \mathcal{S} . Precisely, M is a matrix in which each row corresponds to a possible set \mathcal{R} of random coins for \mathcal{P} , and each of the \mathcal{C}^n columns corresponds to a value of the challenge $\boldsymbol{x} = (x_1, \ldots, x_n)$. The matrix contains a one in the *i*-th row and the *j*-th column if \mathcal{P} makes \mathcal{V} accept for the *i*-th set of random coins and the *j*-th challenge, and a zero otherwise. Therefore, the proportion of ones is ε . First, we organize the columns of the matrix by blocks:

$$M = [M_1|\ldots|M_{\mathcal{C}^{n-1}}],$$

where each block corresponds to a fixed value of the n-1 coordinates x_2, \ldots, x_n , *i.e.*, each block has C columns, each corresponding to a value of x_1 .

Now finding the first pair x_1, x'_1 is just finding two ones in the same block and the same row of the matrix. We now specify the rewinding algorithm:

Choose a random entry in the matrix M, and repeat until finding a one.
 Run in parallel two algorithms:

- (a) Choose a random entry in the same block and row as the first one found, and repeat until finding a one.
- (b) Flip a coin that has probability ε/w of coming out heads until heads comes out.

The algorithm ends when either (a) or (b) ends.

Fig. 11. Algorithm TryExtract

Lemma 2. Suppose $\varepsilon \ge 4/C$. Set w = 13, then the algorithm TryExtract presented in Figure 11 has an expected running time $\mathbb{E}[T] \le 14/\varepsilon$ and finds two ones in the same combination of block and row with probability $p \ge 1/8$.

Proof. Let us first study the algorithm Algo_a of line (a). We consider the event E: "the one found in step 1 is in a combination block/row with a proportion at least $\varepsilon/2$ of ones". In this case, the number of other ones in the same block/row is at least $\mathcal{C}\varepsilon/2-1$, so each trial to find another one has a probability of success of at least $\frac{C\varepsilon/2-1}{C}$. Its expected running time is

$$\mathbb{E}(T_a \mid E) \le \frac{\mathcal{C}}{\mathcal{C}\varepsilon/2 - 1} \cdot$$

As soon as $\varepsilon \geq 4/\mathcal{C}$, one has $\mathbb{E}[T_a \mid E] \leq 4/\varepsilon$. In particular, by Markov inequality,

$$\mathbb{P}(T_a \ge 8/\varepsilon \mid E) \le 1/2.$$

Moreover, at least half of the ones are in a row/block with a proportion at least $\varepsilon/2$ of ones, *i.e.*, $\mathbb{P}(E) \ge 1/2$.

The algorithm of line (b) is a succession of identical trials with probability of success $\varepsilon/13$. The probability that the first success happens after the *m*-th trial is

$$\mathbb{P}(T_b \ge m) = (1 - \varepsilon/13)^{m-1}$$

In particular, one can show that $\mathbb{P}(T_b \geq 8/\varepsilon) \geq 1/2$. Finally,

$$\begin{split} \mathbb{P}(\mathsf{Algo}_a \text{ ends}) &\geq \mathbb{P}(\mathsf{Algo}_a \text{ ends } \wedge E) \\ &\geq \mathbb{P}(\mathsf{Algo}_a \text{ ends } \mid E)/2 \\ &\geq \mathbb{P}((T_a \leq 8/\varepsilon) \ \wedge (T_b \geq 8/\varepsilon) \mid E)/2 \\ &\geq P(T_a \leq 8/\varepsilon \mid E) \cdot P(T_b \geq 8/\varepsilon)/2 \\ &\geq 1/8. \end{split}$$

Moreover, the expected running time for step 2 is at most the expected running time of (b), so $13/\varepsilon$, and the expected running time of step 1 is $1/\varepsilon$ (the time to find a one in a matrix with proportion ε of ones), so that the expected running time of the algorithm of Figure 11 is at most $14/\varepsilon$.

Lemma 3. If there is a prover $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ with probability $\varepsilon \geq 4/\mathcal{C}$ of making the verifier accept the proof on a statement \mathcal{S} , then there exists an algorithm running in expected time

$$\mathbb{E}[T] \le \frac{112n}{\varepsilon}$$

that outputs n pairs $(\mathbf{x}_1, \mathbf{x}'_1), \ldots, (\mathbf{x}_n, \mathbf{x}'_n)$ such that for every $i \in [\![1, n]\!]$, \mathbf{x}_i and \mathbf{x}'_i differ only by their *i*-th coordinate.

Moreover, there exist n sets of prover's random coins $\mathcal{R}_1, \ldots, \mathcal{R}_n$ such that for all i, $(\mathcal{P}_1(\mathcal{S}, \mathcal{R}_i), \mathbf{x}_i, \mathcal{P}_2(\mathcal{S}, \mathcal{R}_i, \mathbf{x}_i))$ and $(\mathcal{P}_1(\mathcal{S}, \mathcal{R}_i), \mathbf{x}'_i, \mathcal{P}_2(\mathcal{S}, \mathcal{R}_i, \mathbf{x}'_i))$ are two accepting transcripts.

Proof. We set w = 13 and define the following algorithm: for $i \in [\![1, n]\!]$, we define $M_i = [M_{i,1}| \dots |M_{i,C^{n-1}}]$ to be the matrix M in which we reorganized the columns so that each block $M_{i,j}$ corresponds to a given value of $(x_k)_{k\neq i}$. We repeat the algorithm TryExtract until it finds two ones in the same row of a same block, which corresponds to a pair x_i, x'_i (and a prover's randomness \mathcal{R}_i). Given that the probability of success of TryExtract is at least 1/8, the expected number of repetitions between two successes is 8, and so the expected number of instances to find all the n pairs is 8n. This algorithm runs in expected time

$$\mathbb{E}[T] \le \frac{112n}{\varepsilon}.$$

We underline that in Lemma 3, the prover's random coins associated with the accepted transcripts is different for each couple of challenges. This type of rewinding is sufficient for extraction whenever the different components of the challenges do not "interact" in the argument of knowledge. The proof presented in Fig. 1 is such an example: each ciphertext is associated to only one component of the challenge, and the relation proved can be proved independently on each ciphertext.

However, in a case in which the components interact (as in the proof of Fig. 2), we need a stronger property. We now construct an algorithm that is able to find in polynomial time n pairs of challenges $(\boldsymbol{x}_1, \boldsymbol{x}_1'), \ldots, (\boldsymbol{x}_n, \boldsymbol{x}_n')$ as

before, except that there exists a unique prover's random coins that make these challenges accepting.

In the matricial modelization, the first situation corresponds to finding two ones in a same row and in a same block, and repeating the operation n times; while the second situation corresponds to finding 2n ones in a same row, that have to be distributed in a particular way. Concretely, the algorithm first looks for a one in the matrix which fixes a row, and from this point, restricts its browsing to only this row, hoping that the chosen row contains *enough* ones.

Lemma 4. If there is a prover $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ that makes the verifier accept the proof on a statement S with probability $\varepsilon \geq 8/C$, then there exists an algorithm running in expected time

$$\mathbb{E}(T) \le \frac{4(1+390\,n\log n)}{\varepsilon}$$

that outputs n pairs of challenges $(\mathbf{x}_1, \mathbf{x}'_1), \ldots, (\mathbf{x}_n, \mathbf{x}'_n)$ such that for every i, \mathbf{x}_i and \mathbf{x}'_i differ only by their *i*-th coordinate.

Moreover, there exists a set of random coins \mathcal{R} and 2n answers such that for all i, $(\mathcal{P}_1(\mathcal{S}, \mathcal{R}), \mathbf{x}_i, \mathcal{P}_2(\mathcal{S}, \mathcal{R}, \mathbf{x}_i))$ and $(\mathcal{P}_1(\mathcal{S}, \mathcal{R}), \mathbf{x}'_i, \mathcal{P}_2(\mathcal{S}, \mathcal{R}, \mathbf{x}'_i))$ are two accepting transcripts.

Proof. In Figure 12, we define the algorithm $\mathsf{TryExtractRepeat}$, with integers parameters w and t.

- 1. Choose a random entry in M, and repeat until finding a one. Denote L the row of this one.
- 2. For *i* in $[\![1,n]\!]$, reorganize the columns of *L* as $L_i = [\ell_{i,1}| \dots |\ell_{i,C^{n-1}}]$, such that each $\ell_{i,j}$ corresponds to a possible value for $(x_k)_{k\neq i}$. Repeat *t* times the algorithm TryExtract on the row L_i to find two ones in a same block $\ell_{i,j}$.

Fig. 12. Algorithm TryExtractRepeat

We fix w = 24 and

$$t = \left\lceil \frac{\log\left(1 - \exp\left(-\log(2)/n\right)\right)}{\log(7/8)} \right\rceil.$$

Let E be the following event: "the one found in step 1 is in a row with a proportion at least $\varepsilon/2$ of ones". Then E has probability at least 1/2 of happening. If E happens, then for every $i \in [\![1,n]\!]$, in step 2, the algorithm TryExtract is run on a matrix with a proportion at least $\varepsilon/2 \ge 4/\mathcal{C}$ of ones. Doing the same computations as in the proof of Lemma 3, Algo_a in TryExtract runs in time $T_a \le 16/\varepsilon$ with probability at least 1/2, and Algo_b has probability

$$\mathbb{P}[T_b \ge 16/\varepsilon] = (1 - \varepsilon/w)^{|16/\varepsilon| - 1} \ge 1/2$$

of exceeding this time. So TryExtract succeeds with probability at least 1/8, and runs in expected time

$$\mathbb{E}[T_1] \le \frac{2}{\varepsilon} + \frac{w}{\varepsilon} = \frac{2+w}{\varepsilon}$$

Now, if we repeat this t times, the probability that at least one of the instances of TryExtract succeeds is $p_i \ge 1 - (7/8)^t$. We do this for every $i \in [\![1, n]\!]$, so there is a probability

$$p \ge \left(1 - (7/8)^t\right)^n,$$

that for every i, at least one of the t instances succeeds.

Our choice of t ensures that $p \ge 1/2$. This means that knowing that E happens, the probability that we find a pair x_i, x'_i for every $i \in [\![1, n]\!]$ is at least one half. So the probability of success of TryExtractRepeat is

 $\mathbb{P}[\mathsf{TryExtractRepeat succeeds}] \geq \mathbb{P}[E] \times p \geq 1/4.$

Moreover, the expected running time of TryExtractRepeat is

$$\mathbb{E}[T_2] \le \frac{1 + nt(2+w)}{\varepsilon}$$

Finally, consider the algorithm that repeats TryExtractRepeat until it is successful. The expected number of repetitions is 4, and its expected running time is

$$\mathbb{E}[T] \le \frac{4(1 + nt(w+2))}{\varepsilon}$$

One can show that $t \leq 15 \log n$, for n > 1, which gives the bound on the expected running time.