Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments

Geoffroy Couteau[1], Michael Klooß[2], Huang Lin[3], and Michael Reichle[4,5]

[1] CNRS, IRIF, Université de Paris, France, couteau@irif.fr
[2] Karlsruhe Institute for Technology, michael.klooss@kit.edu
[3] Mercury's Wing and Suterusu Project, huanglinepfl@gmail.com
[4] DIENS, École normale supérieure, CNRS, PSL University, 75005 Paris, France
[5] Inria, Paris, France, michael.reichle@ens.fr

Abstract. We introduce a new approach for constructing range proofs. Our approach is modular, and leads to highly competitive range proofs under standard assumption, using less communication and (much) less computation than the state of the art methods, without relying on a trusted setup. Our range proofs can be used as a drop-in replacement in a variety of protocols such as distributed ledgers, anonymous transaction systems, and many more, leading to significant reductions in communication and computation for these applications. At the heart of our result is a new method to transform any commitment over a finite field into a commitment scheme which allows to commit to and efficiently prove relations about *bounded integers*. Combining these new commitments with a classical approach for range proofs based on square decomposition, we obtain several new instantiations of a paradigm which was previously limited to RSA-based range proofs (with high communication and computation, and trusted setup). More specifically, we get:

- Under the discrete logarithm assumption, we obtain the most compact and efficient range proof among all existing candidates (with or without trusted setup). Our proofs are 12% to 20% shorter than the state of the art Bulletproof (Bünz et al., IEEE S&P '18) for standard choices of range size and security parameter, and are more efficient (both for the prover and the verifier) by more than an order of magnitude.
- Under the LWE and SIS assumptions, we obtain range proofs that improve over the state of the art in a batch setting when at least a few dozen range proofs are required.[6]
- Eventually, under reasonable assumptions in class groups, we obtain the first concretely efficient standard integer commitment scheme (without bounds on the size of the committed integer) which does not assume trusted setup.[7]

Keywords. Range Proof, Integer Commitments.

[6] **Changelog** w.r.t. eprint version from 2021-04-27: Fixed calculation of the lattice range proof size.
[7] **Changelog** w.r.t. eprint version from 2021-06-22: Changed class group commitments from Pedersen to ElGamal. See appendix B for details.

Table of Contents

1 Introduction

In this work, we develop new techniques to construct range proofs, an important building block in a variety of modern cryptographic protocols such as distributed ledgers, anonymous transactions, e-cash, e-voting, and many more. The range proofs obtained with our methods are highly competitive with the state of the art: they rely on standard assumptions, require less communication and computation, and do not assume any trusted setup. Furthermore, our approach is modular and can be instantiated in the discrete logarithm setting, in the lattice setting (leading to the most efficient post-quantum range proofs in a batch setting), and in the class group setting. Below, we review some background.

Range proofs and anonymous transactions. Zero-knowledge proofs, introduced in the seminal work of Goldwasser, Micali, and Rackoff [GMR89], allow a prover to convince a verifier that a statement is true, while concealing all information beyond the truth of the statement. They are a fundamental primitive in cryptography, with inumerable applications. Range proofs, whose genesis can be traced back to [Bri+88], are a particular type of zero-knowledge proof where the prover wishes to convince the verifier that a committed value belongs to a certain range. Range proofs are a core building block in numerous applications such as anonymous credentials [Cha90], e-voting [Gro05], and e-cash [CHL05]. Furthermore, efficient range proofs have recently become central components in distributed ledgers, the prime example being the recent integration of Bulletproof [Bün+18] in the cryptocurrency Monero[8] and later Mimblewimble-based anonymous cryptocurrencies such as Beam[9] and Grin[10]. Range proofs also play an essential role in anonymous payment schemes for smart contract platforms such as Zether [Bün+20].

In most of these anonymous payment schemes, (positive and negative) integers are encoded as finite field elements, and negative spendings constitute a valid transaction in general, if they are not explicitly disallowed. This feature can be exploited to launch a double-spending attack, allowing the adversary to print money out of thin air [MIO18]. In a confidential payment scheme where both inputs and outputs of a transaction are hidden in either a digital commitment (as in Monero) or an encryption (as in Zether), range proofs are necessary to guarantee that the hidden value falls into the correct range and prevent the aforementioned overflow attack.

The maximum throughput of a distributed ledger protocol is mainly determined by the maximum block size and average transaction size [Cro+16]. The smaller the transaction size is, the larger the maximum throughput is. The average transaction size in an anonymous payment scheme is largely determined by the zero-knowledge range proof size. Therefore, the proof size is a crucial parameter for the design of a range proof scheme. The proof generation and verification time are also vital to the performance of the system built on the range proof scheme. In the case of a decentralized anonymous payment scheme, the proof generation time will determine how fast the anonymous payment can be launched and have a direct impact on the user experience and system scalability [Cec+17]. The proof verification time, on the other hand, has a great impact on the workload of the miners.

1.1 Standard Approaches for Building Range Proofs

Due to their wide variety of applications, many constructions of range proofs have been proposed over the past decades. All these constructions can be categorized in two main high level approaches, which we outline below.

First method: n-ary decomposition. The first method is the one employed both in the early (folklore) constructions of range proofs, as well as in the latest state-of-the-art constructions (such as Bulletproof). To prove that a committed integer x belong to an interval of the form $[0, n^\ell - 1]$, where n is some small value, this method uses the following high-level template:

[8] `https://web.getmonero.org/resources/moneropedia/bulletproofs.html`

[9] `https://github.com/BeamMW/beam`

[10] `https://cointelegraph.com/news/cryptocurrency-grin-follows-through-with-anticipated-july-17-mainnet-hardfork`

1. First, commit to the n-ary decomposition of x, denoted $(x_0, \cdots, x_{\ell-1})$.
2. Second, prove that the relation $x = \sum_{i=0}^{\ell-1} x_i \cdot n^i$ holds.
3. Third, prove that each component of the committed tuple belongs to $[0, n-1]$. Since n is typically very small, this can be achieved using some brute-force method (for example, when using binary decomposition, it amounts to proving that each component is a bit, which can be done using standard methods).

When the commitment scheme satisfies some homomorphic properties, it is generally simple to lift a proof as above to a proof for a more general interval $[a, b]$. The first instance of this approach is a folklore discrete-logarithm-based construction using the Pedersen commitment scheme to commit to the bit decomposition of x. Denoting $\beta = \log(b-a)$ the bitlength of the interval size and λ the bitlength of group elements, This leads to a range proof communicating $\mathcal{O}(\lambda \cdot \beta)$ bits. This approach was first improved in [CCs08] to $\mathcal{O}(\lambda \cdot \beta / \log \beta)$ by using decomposition in a larger basis, and later in [Gro11] to $\mathcal{O}(\lambda \cdot \beta^{1/3})$, using pairings.

In a recent breakthrough work, the authors of [Bün+18] introduced Bulletproof, which managed to reduce the communication to $\mathcal{O}(\lambda \cdot \log \beta)$ under the plain DLOG assumption (without pairings) while still remaining computationally efficient. Their approach relies on generalized Pedersen commitment to commit to the entire bit-decomposition of x using few group elements, and on a clever recursive proof strategy to simultaneously prove that all committed values are bits.[11] This comes at the cost of a larger number of rounds $\mathcal{O}(\log \beta)$ (but this is typically not a concern in real-world applications, where the Fiat-Shamir heuristic is used to make the proof non-interactive) and a computational soundness guarantee (leading to a zero-knowledge argument instead of a proof).

A strong advantage of the proofs obtained in this line of work is that they do not require any trusted setup. In real-world applications such as cryptocurrencies, this is an important feature to avoid having to trust any central authority with the secure generation of the parameters (we will discuss this more later). Due to this feature and its good concrete efficiency, Bulletproof is currently considered the state of the art method for range proofs, and has found its way into several real-world protocols.

Second method: square decomposition. The second method can be traced back to the work of Boudot [Bou00], and was initially introduced to avoid the large $\mathcal{O}(\lambda \cdot \beta)$ cost of the range proofs obtained (at the time) by the first method. It relies on the following high-level template (or a close variant thereof): first, proving that $x \in [a, b]$ reduces to proving that $x - a$ and $b - x$ (whose commitments can typically be computed homomorphically from a commitment to x) are positive. Now, to prove that a committed value y is positive:

1. First, decompose y as $y = \sum_{i=1}^{4} y_i^2$ over the integers. Lagrange's four square theorem guarantees that such a decomposition exists, and efficient algorithms allow to quickly find one.
2. Second, commit to the y_i and prove (using standard methods) that $y = \sum_{i=1}^{4} y_i^2$ *over the integers*.

The advantage of this method is that it requires committing only to a constant number of components (independent of the interval size), instead of $\approx \beta$ components with the first method. This typically leads to proofs with communication $\mathcal{O}(\beta + \lambda)$ bits. However, it is crucial for this method that the relation is proven *over the integers*: standard commitment schemes such as Pedersen only allow committing values over $\mathbb{Z}_p$ for some prime p, but finding a 4-square decomposition over $\mathbb{Z}_p$ does not provide any guarantee of positivity. Hence, a core component of this line of work is the notion of *integer commitment schemes*, introduced in [FO97; DF02], which allows to commit and prove relations among values directly over the integers.

The square decomposition method has been refined in [Lip03]. Later, the work of Groth [Gro05] observed that one can instead decompose $4y + 1$ as a sum of three squares (positive integers congruent to 1 modulo 4 can always be decomposed this way) to reduce the proof size, and further efficiency and security improvements were described in [CPP17]. A common issue of all these works

[11] There have been several recent follow up works [HKR19; AC20] to Bulletproof, which expand the set of relations captured by the framework, but do not translate into concrete improvements on the size of the range proofs produced by this framework.

is that all known integer commitment schemes require the use of RSA groups or class groups with a hard-to-factor discriminant. This means that the group size is very large (typically 2048 or 3072 bits), and that these proofs all require a trusted setup to generate a public product of secret prime factors[12]. Assuming a trusted setup is a rather undesirable property in a decentralized anonymous payment scheme: in general, the party responsible for the setup step can exploit the trapdoor information obtained through this process to print an unlimited amount of cryptocurrency without being detected [Sle; Ben]. Although one could potentially mitigate the risk of the above attack by using secure multi-party computation to execute the setup step (as was done e.g. for zcash[13]), it introduces additional engineering complexity and potential vulnerabilities.

Furthermore, even before Bulletproof, these proof systems were competitive with proofs obtained with the first method only for very large intervals. Compared to Bulletproof, they lead to much larger proof sizes for any interval size (and are also computationally less efficient). Due to their higher cost and their need of a trusted setup, this second method is largely considered obsolete and non-competitive with the proofs obtained through the first method.

1.2 Our Contribution

In this work, we turn the tables and demonstrate that the square decomposition method can be refined to create highly competitive range proofs, with smaller communication and computation compared to the state of the art Bulletproof, without trusted setup (meaning that our proofs only require a transparent setup), and under standard assumptions. Among other advantages, our method is modular and can also be instantiated in the lattice setting to obtain post-quantum range proofs which are highly competitive with the sate of the art in a batch scenario (where several range proofs must be computed at once), and in the class group setting with prime discriminant. Furthermore, our proofs require only three rounds of interaction, an important feature if one does not want to rely on the Fiat-Shamir heuristic, and can be modified to achieve statistical soundness instead of computational soundness (at a small cost in efficiency). At the heart of our constructions is a new generic method to convert any commitment scheme over $\mathbb{Z}_p$ into a *bounded integer commitment scheme*, i.e., a commitment scheme which allows to commit to bounded-range integers and to prove relations over $\mathbb{Z}$ between committed bounded-range integers.

Instantiation in the discrete-log setting. Instantiating our framework with the standard Pedersen commitment scheme, we obtain a bounded integer commitment scheme under the discrete logarithm assumption. When plugging this bounded integer commitment scheme in the range proof of [CPP17], we obtain a range proof which does not require any trusted setup and can benefit simultaneously from the compactness of square-decomposition-based range proofs (i.e., constant number of group elements) and the possibility of instantiating the Pedersen commitment scheme over prime-order elliptic curve, with small group elements[14]. To further optimize the proof size, we describe an optimized variant which relies on the *short-exponents* discrete logarithm assumption (i.e., the assumption that it is hard to compute discrete logarithm even when the exponent is sampled from a large enough bounded range), which is a well-studied variant of the standard discrete log assumption. For example, for an interval size of 2^{32} and 128 bits of security, we obtain range proofs of size 501 Bytes, compared to the 608 Bytes of Bulletproof. For the same parameters, the computational cost for both the prover and the verifier are more than an order of magnitude smaller compared to Bulletproof. The high efficiency of prover *and* verifier is crucial for use of (range) proofs on resource constrained devices, such as smartphones. Such devices are of special interest for privacy-enhancing technologies, such as anonymous credentials [Cha90] and payment systems. To achieve practicalility, tradeoffs have to made. For example, the work [Blö+19] relies on [CCs08],

[12] While it is theoretically possible to use a very large random integer as RSA modulus, without relying on a trusted party to compute a product of safe primes, this approach is completely impractical due to the very large group size and amount of computation, see the discussion on RSA-UFO in [LM19].

[13] `https://z.cash/technology/paramgen/`

[14] Since our bounded integer commitment scheme requires the committed values to remain into a bounded range, we actually require slightly larger group size compared to Bulletproof to achieve the same security level; this is accounted for in our concrete comparison and will be covered in details in the technical overview.

which requires pairings and relatively large public parameters, whereas the work [Hof+20] relies on *uncompressed*, i.e. linear-size, Bulletproofs, trading communication for computation. Our range proofs are a great fit for these settings.

Detailed comparison with Bulletproof. A more detailed comparison with Bulletproof is given in Table 1. Below, we explain how the numbers in the table have been obtained. Computing the exact costs of our range proof is rather tedious, since it involves careful optimizations with rejection sampling techniques, and optimizations using the short-exponent discrete logarithm assumption. We consider range proofs over an interval $[a, b]$ with $\beta = \log(b - a) \in \{32, 64\}$, a security parameter $\lambda \in \{80, 128\}$, and a group of size q (which might not be the same for Bulletproof and our range proof). The formula below additionally uses parameters C, S, L' corresponding respectively to the challenge size, a bound on the length of short exponents, and a bound for rejection sampling. Our concrete numbers are obtained by setting $C = 2^\lambda, S = 2^{2\lambda}, L' = \lceil 256\sqrt{2\lambda} \rceil$. The formulas for computing the range proof size (in the non-interactive setting, when Fiat-Shamir is used), the prover work, and the verifier work, are given below:

- Proof size (in bits): $30(\beta + \log(CL')) + \lceil \log(C)/\lambda \rceil (2\lambda + 4(2\beta + \log(CL') + 2\log(SCL')) + 2)$ (our work) versus $\log q \cdot (2\beta + 9)$ (Bulletproof).
- Prover work (in group multiplications): $2.31 \cdot (4\beta + 8\log S + 6\log C + 7\log L') + 30$ (our work) versus $18 \cdot (\beta \log q)$ (Bulletproof).
- Verifier work (in group multiplications): $4.5\beta + 7\log S + 13\log C + 9\log L' + 10$ versus at least $3\beta \cdot \log q$ (lower bound on the cost for Bulletproof, computed as the cost of a single inner product argument)
- Group size (in bits): $\log q = \log(32(2^\beta CL')^2) + 1$ (our work) versus $\log q = 2\lambda$ (Bulletproof)

In the above, prover and verifier work are computed as the number of multiplications required for the exponentiations (we do not directly count the exponentiations for fairness of comparison: Bulletproof and our work do not use the same group size, and our optimized construction also uses exponentiations with short exponents), which largely dominate the overall cost. We note that in both our work and Bulletproof, the verifier work can be optimized by relying on multiexponentiations techniques; since these techniques apply identically in both works and do not significantly change the bottom line in terms of comparisons, we ignore them in this overview.

Asymptotically, our proofs have size $O(\lambda+\beta)$, while Bulletproof has size $O(\lambda \log \beta)$. We note that in the range of parameters $\beta = O(\lambda)$, our techniques actually lead to an asymptotic improvement over Bulletproof; for larger ranges, Bulletproof is more efficient, and for very small ranges, the asymptotic costs are the same for both. Previous square-decomposition-based range proofs had asymptotic cost $O(\beta + \lambda^{3-o(1)})$ due to their use of RSA modulus (which allow for subexponential attacks).

We stress that when not using the Fiat-Shamir heuristic, our scheme can be instantiated to have only three rounds (this slightly increases the proof size, because it requires to not use rejection sampling, since the latter causes the protocol to restart with non-negligible probability) while Bulletproof requires $\log \beta$ rounds. Even with rejection sampling and our concrete choice of parameters, the expected number of rounds is less than 5. The round complexity is known to strongly impact the tightness of the security loss in the random oracle model. Thus for sufficiently large β, our security analysis is significantly tighter than the one of Bulletproofs in the random oracle model.

Furthermore, our scheme can be instantiated to have statistical soundness. On the other hand, Bulletproof allows for extremely efficient batching of a large number of range proofs communication-wise, and would therefore become preferable when many range proofs must be performed at once if communication is the sole concern (though it usually is not). In any case, and independently of the number of range proofs, our range proofs requires 20 to 40 times less group multiplications for the prover, and 6 to 15 times less for the verifier.

Instantiation in the lattice setting. For the instantiation of our framework in the lattice setting, we build upon the commitment scheme and proof system from [Yan+19]. The commitments built this way allow to commit to long vectors over $\mathbb{Z}_q^n$ (think of n as being a few thousands, e.g.

Table 1. Comparison between the optimized range proof of Section 5.4 and Bulletproof [Bün+18] for various choices of security parameter λ and log of interval size β. Proof size and group size are in Bytes, prover and verifier work are counted as a number of group multiplications, rounded to two decimal places. See the paragraph "detailed comparison with Bulletproof" for the details on our computations.

(β, λ)		proof size	prover work	verifier work	Group size
$(32, 80)$	This Work	339	4.6k	2.4k	32
	Bulletproof	380	92k	$>$ 15k	20
$(32, 128)$	This Work	501	7k	3.7k	44
	Bulletproof	608	150k	$>$ 25k	32
$(64, 80)$	This Work	383	4.9k	2.6k	40
	Bulletproof	420	180k	$>$ 31k	20
$(64, 128)$	This Work	545	7.3k	3.8k	52
	Bulletproof	672	290k	$>$ 49k	32

$n = 5000$). Our techniques require to use a relatively large modulus q in order to avoid overflows in the computation. As a consequence, our commitments and proofs are quite large.

However, in exchange for using a large modulus, the commitment and proof system obtained by compiling the commitment of [Yan+19] with our techniques allow to *batch* many range proofs extremely efficiently: we can essentially perform up to n range proofs in parallel for the cost of a single range proof, even if the range proofs have different ranges. This improves over the communication achieved by the best plain LWE-based range proofs [Yan+19] in a batch setting. For example for a total proof size of 1.21 MB and $\beta = 32$, the scheme from [Yan+19] allows to batch 32 range proofs at once, wheras our scheme allows for 180 range proofs.

Note that under Ring-LWE (or Module-LWE) assumptions, range proofs ([Esg+19; Boo+20]) obtain better efficiency. In appendix A, we discuss the application of our techniques in this setting.

Instantiation in the class group setting. Eventually, we also instantiate our method in the class group setting. The proofs obtained this way improve over our DLOG-based proofs only for large ranges, where Bulletproof would be more efficient. On the other hand, instantiating our approach in the class group setting leads to the first concretely efficient construction of *unbounded* integer commitment scheme which does not require a trusted setup (the only known alternative uses RSA-UFO, which is impractical, see the discussion in [LM19]).

Concurrent Works. In the DLOG setting, the work of [Chu+20] recently claimed an improvement in proof size compared to [Bün+18] by slightly reducing the number of group elements required in [Bün+18]. The computational cost of their proof is the same as in [Bün+18]. To our knowledge, their scheme was not peer reviewed yet; we note that our range proofs are still shorter than theirs, and more than an order of magnitude computationally more efficient.

2 Technical Overview

As we outlined in the introduction, at the heart of our results is a method to convert standard homomorphic commitment schemes into *bounded integer commitment schemes* – that is, a scheme that allows to commit to integers from a bounded range, but also to *prove in zero-knowledge* relations between commited values *over the integers*, see [FO97; DF02] – with a certain set of additional specific properties. We now provide details on our approach.

2.1 A Natural Approach via Σ-Protocols

For simplicity, suppose that we have at our disposal a commitment scheme com with message space and random coin space $\mathbb{Z}_q$, for some large prime q, which is homomorphic over the messages and

the coins: $\mathsf{com}(m_1; r_1) \cdot \mathsf{com}(m_2; r_2) = \mathsf{com}(m_1 + m_2; r_1 + r_2)$. This is satisfied for example by the Pedersen commitment scheme $\mathsf{com}(m; r) = g^m h^r$ for two group elements (g, h) over a group of order q. The transformation works for a more general class of commitments, this choice of structure is for the sake of concreteness in the presentation. Suppose now that we would like to obtain a bounded integer commitment scheme out of com. The first obvious idea is to proceed as follows:

- map values in $\mathbb{Z}_q$ to integers $[-(q-1)/2, (q-1)/2]$ in the natural way;
- define com' to be exactly like com, but where the committed values are restricted to $[-R, R]$, where $R \ll (q-1)/2$ is some bound.

Intuitively, the bound R is here to ensure that we will have enough "room" to guarantee that if a relation between elements of $[-R, R]$ holds modulo q, then it must also hold over the integers. Looking ahead, for building a range proof, we will want to prove relations of the form $x = \sum_i x_i^2$, and we will choose R such that no overflow occurs when computing $\sum_i x_i^2 \bmod q$ with $x_i \in [-R, R]$.

The next step is to equip this commitment com' with a zero-knowledge proof system allowing to prove relations between committed values *over the integers*. However, this turns out to be particularly challenging. To see this, consider the standard Σ-protocol between a prover P and a verifier V for proving knowledge of an opening (m, r) to a commitment $c = \mathsf{com}(m; r)$:

- P: pick $(m', r') \xleftarrow{\$} \mathbb{Z}_q^2$ and send $c' = \mathsf{com}(m'; r')$.
- V: send a challenge $e \xleftarrow{\$} \mathbb{Z}_q$.
- P: send $d_m = em + m'$ and $d_r = er + r'$.
- V: accept if $\mathsf{com}(m; r)^e \cdot \mathsf{com}(m'; r') = \mathsf{com}(d_m; d_r)$.

Using a standard rewinding argument, we can extract a valid opening $(m; r) \in \mathbb{Z}_q^2$ of c from any (potentially malicious) prover P^* which produces accepting proofs with non-negligible probability ε: run P^* to get c', fork it, and run it on two different random challenges e, e', receiving (d_m, d_r) and (d'_m, d'_r). By a standard probability lemma (see the splitting lemma from [PS96; PS00]), (c', e, d_m, d_r) and (c', e', d'_m, d'_r) will both be accepting transcript with non-negligible probability $\Omega(\varepsilon^2)$. From the two accepting equations, one gets

$$c = \mathsf{com}((d_m - d'_m) \cdot (e - e')^{-1}, (d_r - d'_r) \cdot (e - e')^{-1}). \tag{1}$$

To adapt the protocol to com', we would need to modify the Σ-protocol such that it additionally guarantees that the extracted value m belongs to $[-R, R]$. This actually seems feasible at first sight if we agree to settle for a relaxed correctness and zero-knowledge guarantee: we only enforce correctness and (honest-verifier) zero-knowledge whenever m belongs to $[-R', R']$, for a bound R' such that $2^{\lambda+\kappa} R' \leq R$, where κ is a statistical security parameter for zero-knowledge, and λ is a statistical security parameter for soundness (we keep both separate for generality). Then, we can modify the protocol as follows:

- P: pick $(m', r') \xleftarrow{\$} [-2^{\lambda+\kappa} R', 2^{\lambda+\kappa} R'] \times \mathbb{Z}_q$ and send $c' = \mathsf{com}(m'; r')$.
- V: send a challenge $e \xleftarrow{\$} [1, 2^\lambda]$.
- P: send $d_m = em + m'$ and $d_r = er + r'$.
- V: accept if $\mathsf{com}(m; r)^e \cdot \mathsf{com}(m'; r') = \mathsf{com}(d_m; d_r)$ and $d_m \in [-R, R]$.

Intuitively, relaxed correctness and relaxed statistical zero-knowledge follow from the fact that for $m \in [-R', R']$ and $e \in [1, 2^\lambda]$, $d_m = em + m'$ for $m' \xleftarrow{\$} [-2^{\lambda+\kappa} R', 2^{\lambda+\kappa} R']$ will be $2^{-\kappa}$-close to uniform (in statistical distance) over $[-R, R]$. It remains to analyze whether we can extract from an accepting prover a valid witness for com'. However, even though we restricted e and d_m to be small, recall that the extracted value (Equation 1) is of the form $m = (d_m - d'_m) \cdot (e - e')^{-1} \bmod q$. That is, m is not an element of $[-R, R]$ in general; rather, it is the product of an element in $[-R, R]$ and *the inverse modulo q of an element in* $[1, 2^\lambda]$. Therefore, this approach fails at binding the prover to a value $m \in [-R, R]$.

We note that the failure of this approach – the impossibility of extracting values guaranteed to be short in general – is a well-known problem in the context of lattice-based cryptography. Indeed, standard Σ-protocol for proving knowledge of a short solution to a system of equation – i.e., a witness for the SIS problem – suffer from exactly the same limitation (see e.g. the discussions

in [Ben+14]). The standard solution is to restrict the challenge set to $\{0,1\}$ (to guarantee that the inverse of the difference between distinct challenges remains small), and to amplify soudness via parallel repetitions. However, in our context, this would lead to a very inefficient proof system. Unfortunately, finding a different proof system with much better efficiency seems to be a hard problem (especially in the single-proof setting).

2.2 Encoding Integers as mod-q Rationals

Instead, we follow a different approach by turning the problem around: rather than searching an efficient and sound proof system for the commitment com' above, we seek to find a different construction of bounded integer commitment $\overline{\mathsf{com}}$ such that the above efficient proof system – which is not sound because it only allows extracting fractions of small values modulo p – becomes a sound proof system for $\overline{\mathsf{com}}$ (allowing to extract bounded integers committed with $\overline{\mathsf{com}}$). Abstracting out, we saw above that we can extract from a cheating prover a triple $(y, d, \rho) \in [-R, R] \times [1, 2^\lambda] \times \mathbb{Z}_q$ such that $c = \mathsf{com}(y \cdot d^{-1} \bmod q; \rho)$. Our goal will be to find an appropriate choice of *encoding* Encode satisfying the following properties:

- $\overline{\mathsf{com}}(x; \rho) = \mathsf{com}(\mathsf{Encode}(x); \rho)$, such that a commitment to a value x' with com can be seen as a commitment to some different value $x = \mathsf{Decode}(x')$ with $\overline{\mathsf{com}}$.
- Extracting a tuple $(y, d, \rho) \in [-R, R] \times [1, 2^\lambda] \times \mathbb{Z}_q$ should correspond to extracting a valid opening of $\overline{\mathsf{com}}$ to some *bounded integer* x in an appropriate bounded range.

Looking ahead, we will need a few additional properties to hold for Encode if we want to build an efficient range proofs for $\overline{\mathsf{com}}$.

- First, we want Encode to satisfy some appropriate homomorphic properties. Informally, we want: $\mathsf{Encode}(-x) = -\mathsf{Encode}(x)$, $\mathsf{Encode}(x + a) = \mathsf{Encode}(x) + a$, and $\mathsf{Encode}(a \cdot x) = a \cdot \mathsf{Encode}(x)$, for a sufficiently small integer a.
- Second, we want to be able to transfer a square decomposition from encodings modulo q to encoded integers: informally, proving a relation of the form $x' = \sum_i (x'_i)^2 \bmod q$ where $x' = \mathsf{Encode}(x)$ and $x'_i = \mathsf{Encode}(x_i)$ should guarantee that $x = \sum x_i^2$ *over the integers.*

Our choice of encoding. It turns out that there is a choice of (randomized) encoding that satisfies all of the above constraints simultaneously. In hindsight, this encoding is quite simple and natural: we view any pair $(y, d) \in [-R, R] \times [1, 2^\lambda]$ as an encoding $(y, d) = \mathsf{Encode}(x)$ of the integer

$$x = \left\lfloor \frac{y}{d} \right\rceil \in [-R, R],$$

where the fraction denotes standard division, and $\lfloor \cdot \rceil$ denotes *rounding to the nearest integer*. Given this choice of encoding, $\overline{\mathsf{com}}$ is defined as follows:

- $\overline{\mathsf{com}}(x)$: pick $\rho \xleftarrow{\$} \mathbb{Z}_q$ and output commitment $c = \mathsf{com}(x; \rho)$ and opening $(x, 1, \rho)$.
- $\overline{\mathsf{com}}.\mathsf{Verify}(c, \vec{x}, (y, d, \rho))$: check that $c = \mathsf{com}(y \cdot d^{-1}; \rho)$, $x = \lfloor y/d \rceil$, $y \in [-R, R]$, and $d \in [1, 2^\lambda]$.

Some remarks are in order. First, observe that $\overline{\mathsf{com}}(x)$ is defined exactly as $\mathsf{com}(x)$; that is, an honest commitment with $\overline{\mathsf{com}}$ is just a normal commitment with com. This is because we can view any $x \in [-R, R]$ as an encoding $(x, 1)$ of itself (since $x = \lfloor x/1 \rceil$). The only difference is that we relax the verification to accept general openings $(y, d) = \mathsf{Encode}(x)$ of x. Second, the fact that extracting a triple (y, d, ρ) in the Σ-protocol corresponds to extracting a valid opening (w.r.t. $\overline{\mathsf{com}}$) of an integer in $[-R, R]$ becomes trivially true. It remains to check two things:

1. $\overline{\mathsf{com}}$ must remain *binding* and *hiding*;
2. $\overline{\mathsf{com}}$ must satisfy some homomorphic properties that we outlined above.

$\overline{\mathsf{com}}$ is binding and hiding. That $\overline{\mathsf{com}}$ is hiding follows immediatly from the fact that com is hiding. It remains to consider binding. Suppose that an adversary finds two valid openings (y, d, ρ) and (y', d', ρ') in $[-R, R] \times [1, 2^\lambda] \times \mathbb{Z}_q$ to a commitment c; that is, $c = \mathsf{com}(y \cdot d^{-1} \bmod q; \rho) = \mathsf{com}(y' \cdot (d')^{-1} \bmod q; \rho')$. Since com itself is binding, we must have $y \cdot d^{-1} = y' \cdot (d')^{-1} \bmod q$. This last equation implies

$$yd' = y'd \bmod q \implies yd' = y'd \text{ over } \mathbb{Z} \implies \lfloor y/d' \rceil = \lfloor y'/d \rceil,$$

where the first implication holds as long as q is chosen large enough compared to R and 2^λ, i.e., $q/2 > R \cdot 2^\lambda$.

Properties of $\overline{\mathsf{com}}$. First, we check some basic homomorphic properties:

- If (y, d) encodes $x = \lfloor y/d \rceil$, then $(-y, d)$ encodes $-x$.
- If (y, d) encodes $x = \lfloor y/d \rceil$ and a is an integer such that $ya \leq R$, then $\overline{\mathsf{com}}(x)^a = \mathsf{com}(ayd^{-1})$ is a valid commitment $\overline{\mathsf{com}}(ax)$.
- If (y, d) encodes $x = \lfloor y/d \rceil$ and a is an integer such that $y + da \leq R$, then $\overline{\mathsf{com}}(x) \cdot \mathsf{com}(a) = \mathsf{com}(yd^{-1} + a) = \mathsf{com}((y + da)d^{-1})$ is a valid commitment $\overline{\mathsf{com}}(x + a)$ since $\lfloor (y + da)/d \rceil = \lfloor y/d + a \rceil = \lfloor y/d \rceil + a$.

Second, in our most optimized range proof constructions, we will reduce the task of proving that x belongs to an interval $[a, b]$ to the task of proving that $x_0 = (x - a)(b - x)$ is positive. To show the latter, we will prove that there exists three integers (x_1, x_2, x_3) such that $4x_0 + 1 = \sum_{i=1}^3 x_i^3$; such a decomposition exists (and can be found efficiently) if and only if $x_0 \geq 0$ [Gro05]. Now, suppose we extracted encodings $(y, d), ((y_i, d)_{i \leq 3})$ to $4x_0 + 1$ and (x_1, x_2, x_3) respectively, with the following guarantee: $yd^{-1} = \sum_{i=1}^3 (y_i d^{-1})^2 \bmod q$.

Intuitively, this guarantee will be obtained by using a standard Σ-protocol to prove knowledge of a 3-square decomposition directly over commitments with com. The extracted encodings will all have a common d, because of the structure of the extraction procedure: d corresponds simply to the difference between two distinct challenges for which the prover produced an accepting transcript. The above equation can be rewritten $yd = \sum_{i=1}^3 y_i^2 \bmod q$, which necessarily holds over the integers (i.e., no overflow occurs) given that $3R^2 < q/2$ and $2^\lambda R < q/2$, since the values y and y_i are bounded by R and d is bounded by 2^λ. From there, dividing both sides by d^2 over the rationals, we get that y/d can we written as a sum of three squares over $\mathbb{Q}$. A simple technical lemma shows that this relation over $\mathbb{Q}$ actually suffices to guarantee $x = \lfloor y/d \rceil \in [a, b]$; we omit details in this high level overview.

Note that in related work [FSW03], a similar encoding is used to allow for homomorphic computations with bounded rationals. However in our case, bounded rationals appear as an intermediate result as extracted value $(y - y') \cdot (d - d')^{-1} \bmod q$ of the proof of knowledge. Our encoding is for small integers, hence the rounding. Also, the work [LN17] uses the fact that the extracted value is unique to construct verifiable encryption schemes. Again, the application differs.

2.3 Instantiation in the Discrete Log Setting

Equipped with a method to build bounded integer commitment schemes which satisfy some necessary properties, we turn to the problem of instantiating the construction in different settings, and building a range proof from it. In the discrete logarithm setting, we set com to be the standard Pedersen commitment scheme: $\mathsf{com}(m; r) = g^m h^r$ where (g, h) are two random generators over a group where computing discrete logarithms is hard. As for the range proof, we rely on the efficient Σ-protocol of [CPP17], adapting it to prime order group (since the scheme is described over subgroups of $\mathbb{Z}_n$ for an RSA modulus n in [CPP17]). This is a relatively standard Σ-protocol where the prover, given an opening (x, r) for a commitment $c = g^x h^r$, commits to three values (x_1, x_2, x_3) such that $4(x-a)(b-x)+1 = \sum_i x_i^2$, and proves knowledge of openings to x, x_1, x_2, x_3 such that this relation is satisfied. We provide a detailed security analysis of the resulting protocol.

The scheme of [CPP17] already includes a standard optimization for Σ-protocols, which relies on a collision-resistant hash function to compress the first flow while preserving soundness. We introduce two important additional optimizations tailored to our setting.

First Optimization. Due to our use of a group with a large order, we can actually reduce the size of the random coins used in the Pedersen commitments, at the cost of relying on the *short-exponent* discrete logarithm assumption (DLSE). This improves the computational efficiency, but also reduces the communication when proving knowledge of an opening. Furthermore, relying on DLSE has an important consequence: while the protocol of [CPP17] has computational soundness (and statistical zero-knowledge), we get an alternative instantiation which satisfies statistical soundness (and computational zero-knowledge).

On getting range proofs with statistical soundness. This alternative instantiation is obtained by changing the commitment as follows: To commit to $m \in [-R, R]$, sample $r \xleftarrow{\$} [1, K]$ and output $g^m h^r$. Here, R is a bound on the committed messages, and K is chosen such that the short-exponent discrete log assumption, with random exponent chosen from $[1, K]$, is believed to hold. Applying DLSE, h^r is indistinguishable from a uniformly random group element (using a standard search-to-decision reduction for DLSE in prime-order groups [KK04]). Hence, the scheme remains (computationally) hiding. Furthermore, $g^m h^r$ is perfectly binding: the probability (over the random choice of s such that $g^s = h$) that there exists (m, r, m', r') with $m' \neq m$ such that $m + sr = m' + sr'$ is negligible by the Schwartz-Zippel lemma and a union bound (when R, K are small enough).

Therefore, using our proof system with short randomness in the Pedersen commitments, with appropriate parameter adjustment to guarantee perfect binding, we obtain a range proof with statistical soundness. We note that this is an important feature: the impossibility of getting statistical soundness with Bulletproof is discussed in Section 4.6 of the Bulletproof paper [Bün+18]. In anonymous transaction schemes, statistical soundness is more important than statistical zero-knowledge, since the former is crucial for avoiding indetectable creation of coins (which would render the currency useless), while the second is only necessary to guarantee anonymity (without which the currency remains usable). Not getting statistical soundness was generally believed to be inherent to efficient range proofs, since very compact commitments require computational soundness; our method shows that it is actually possible to get competitive range proofs with statistical soundness. Note that there is also a natural instantiation of our approach using ElGamal encryption as the underlying commitment scheme. This also yields a statistically sound range proof but it is less efficient than the variant of this work.

Second Optimization. The scheme of [CPP17] relies on standard "flooding" to achieve statistical zero-knowledge: the value $e \cdot m$, where $m \in [-R, R]$ is a secret value and $e \leq 2^\lambda$ is a challenge, is masked with a random $m' \xleftarrow{\$} [1, 2^{\lambda+\kappa} R]$ to ensure that $em + m'$ will be $2^{-\kappa}$-close in statistical distance to the uniform distribution over $[1, 2^{\lambda+\kappa} R]$. However, it turns out that our constraints are closely related to the constraints satisfied by several Σ-protocols in the *lattice* setting, which also deal with careful bounds on the size of secret values. Building upon this observation, we import a standard optimization of Σ-protocols in the lattice-setting, namely, the *rejection-sampling* method [Lyu12]. Using rejection sampling allows different tradeoffs between the group size, the number of repetitions of the underlying protocol, and the size of the masks used to hide secret values. We show that an appropriate choice of tradeoff allows to significantly reduce the communication complexity of our protocol.

3 Preliminaries

Notation. In this work, we generally perform calculations in $\mathbb{Z}/q\mathbb{Z}$ with representatives $\mathbb{Z}_q = [-\frac{q-1}{2}, \frac{q-1}{2}]$ for an odd modulus $q \in \mathbb{N}$, and we identify $\mathbb{Z}_q$ with $\mathbb{Z}/q\mathbb{Z}$, unless stated otherwise. Inside of flooring $\lfloor \frac{a}{b} \rfloor$ or rounding $\lfloor \frac{a}{b} \rceil = \lfloor \frac{a}{b} + \frac{1}{2} \rfloor$ operations, we generally have a, b in $\mathbb{Z}$ with division over $\mathbb{Q}$, i.e. we work with the representatives and not in $\mathbb{Z}/q\mathbb{Z}$.

For some randomized algorithm $\mathscr{A}$ with input x, we sometimes write $y \leftarrow \mathscr{A}(x; r)$ for its execution with explicit randomness r. If the randomness is not explicit, we write $y \leftarrow \mathscr{A}(x)$ and assume that the randomness was sampled accordingly. We also write $s \xleftarrow{\$} S$ for sampling s uniformly random from a finite set S or $d \xleftarrow{\$} D$ to sample d randomly according to a given probability distribution D. Further, we often assume that some public parameters, denoted by pp, and the security parameter, denoted by λ, are implicitly passed as input to algorithms if it is clear by context.

A sampling algorithm Sample has *invertible sampling* if, given any output y of $\mathsf{Sample}(1^\lambda, \mathsf{pars})$, one can efficiently compute random coins ρ such that $y = \mathsf{Sample}(1^\lambda, \mathsf{pars}; \rho)$, and the distribution of ρ statistically close to uniform conditioned on output y.

Throughout, we write integers $a \in \mathbb{Z}$ in lower case letters, vectors as $\vec{a} \in \mathbb{Z}^n$ with components a_i, and matrices $\boldsymbol{A} \in \mathbb{Z}^{m \times n}$ in bold upper case letters. Computations on vectors are performed component-wise, unless stated otherwise. For example, for vectors $\vec{a} = (a_i)_{i=1..n}, \vec{b} = (b_i)_{i=1..n} \in \mathbb{Z}^n$ and scalar $y \in \mathbb{Z}$, we write $\vec{c} = \vec{a} \cdot \vec{b} = (a_i \cdot b_i)_{i=1..n}, y^{\vec{B}} = (y^{b_i})_{i=1..n}$ and $\vec{B}^y = (b_i^y)_{i=1..n}$. For some constant $c \in \mathbb{Z}$, we let by $\vec{c} = (c)_{i=1..n}$ the vector with all components equal to c.

We denote by $|x|$ the absolute value of $x \in \mathbb{R}$ and by $\|\cdot\|_1, \|\cdot\|_2, \|\cdot\|_\infty$ the norms defined as $\|\vec{x}\|_1 = \sum_i |x_i|$, $\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$, $\|\vec{x}\|_\infty = \max_i |x_i|$ for $\vec{x} \in \mathbb{R}^m$.

3.1 Hash Functions

A (keyed) hash function H is of the form $\mathsf{H}\colon \mathcal{K} \times \{0,1\}^* \mapsto \{0,1\}^l$. The key (i.e. the first input) to H is usually implicit, and part of the public parameters. We require *keyed* hash functions to achieve collision-resistance against non-uniform adversaries.

Definition 1 (CRHF). Let $\mathsf{H}\colon \mathcal{K} \times \{0,1\} \mapsto \{0,1\}^l$ be a hash function. We call H a *collision-resistant* hash function (CRHF), if for all PPT adversaries $\mathscr{A}$ there exists a negligible function negl such that

$$\Pr\left[k \leftarrow \mathcal{K}; (m_0, m_1) \leftarrow \mathscr{A}(1^\lambda, k)\colon\ m_0 \neq m_1 \wedge \mathsf{H}(k, m_0) = \mathsf{H}(k, m_1)\right] \leq \mathrm{negl}(\lambda).$$

We generally let $l = 2\lambda$ since there are black box attacks (for example generic birthday attacks) that break collision resistance in expected $2^{l/2}$ accesses, so $l \geq 2\lambda$ is necessary for achieving λ bits of security.

3.2 Commitment Schemes

A commitment scheme com with message space $\mathcal{M}_{\mathrm{com}}$, commitment space $\mathcal{C}_{\mathrm{com}}$ and opening space $\mathcal{R}_{\mathrm{com}}$ is a 3-tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ such that

- $\mathsf{com.Setup}(1^\lambda)$: outputs public parameters pp,
- $\mathsf{com.Commit}_{\mathsf{pp}}(x)$: computes a commitment $c \in \mathcal{C}_{\mathrm{com}}$ to $x \in \mathcal{M}_{\mathrm{com}}$ with its opening $d \in \mathcal{R}_{\mathrm{com}}$ and outputs the pair (c, d),
- $\mathsf{com.Verify}_{\mathsf{pp}}(c, x, d)$: verifies the commitment $c \in \mathcal{C}_{\mathrm{com}}$ to $x \in \mathcal{M}_{\mathrm{com}}$ with the opening $d \in \mathcal{R}_{\mathrm{com}}$ and outputs a bit $b \in \{0,1\}$

Further, we require that com statisfies the correctness, binding and hiding properties defined below. Often, d consists of the randomness used in the commitment generation, but it can include other auxiliary information.

Definition 2 (Correctness of a Commitment Scheme). A commitment scheme com is *correct* if for any $\mathsf{pp} \xleftarrow{\$} \mathsf{com.Setup}(1^\lambda)$, any message $m \in \mathcal{M}_{\mathrm{com}}$ and for $(c, d) \leftarrow \mathsf{com.Commit}_{\mathsf{pp}}(m)$, it holds that $\mathsf{com.Verify}_{\mathsf{pp}}(c, d, m) = 1 - \mathrm{negl}(\lambda)$.

Definition 3 (Hiding Property of a Commitment Scheme). A commitment scheme com is *hiding* if for any PPT adversary $\mathscr{A}$, it holds that

$$\Pr\left[\begin{array}{ll} \mathsf{pp} \xleftarrow{\$} \mathsf{com.Setup}(1^\lambda), & (m_0, m_1, \mathsf{st}) \leftarrow \mathscr{A}(\mathsf{pp}), \\ b \xleftarrow{\$} \{0,1\}, & (c, d) \leftarrow \mathsf{com.Commit}_{\mathsf{pp}}(m_b), : b' = b \\ b' \leftarrow \mathscr{A}(\mathsf{st}, c) & \end{array}\right] \leq \frac{1}{2} + \mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$.

Definition 4 (Binding Property of a Commitment Scheme). A commitment scheme com is *binding* if for any PPT adversary $\mathscr{A}$, it holds that

$$\Pr\left[\begin{array}{c} \mathsf{pp} \xleftarrow{\$} \mathsf{com.Setup}(1^\lambda), (c, d_0, d_1, m_0, m_1) \leftarrow \mathscr{A}(\mathsf{pp}) : \\ m_0 \neq m_1 \wedge \mathsf{com.Verify}_{\mathsf{pp}}(c, d, m_0) = \mathsf{com.Verify}_{\mathsf{pp}}(c, d, m_1) = 1 \end{array}\right] \leq \mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$.

(Homomorphic) Integer Commitment Schemes. In this work, we are interested in integer commitment schemes which allow to commit to an integer $x \in \mathbb{Z}$. An integer commitment scheme has message space $\mathcal{M}_{\text{com}} = \mathbb{Z}$ and allows for proving relations, such as knowledge of an opening, in a zero-knowledge manner (see section 3.3). We also establish bounded integer commitment schemes (section 4.1) where the message space is $\mathcal{M}_{\text{com}} = \{x \in \mathbb{Z} \mid |x| \leq R\}$ for some upper bound R. The crucial difference between message space $\mathcal{M}_{\text{com}} = \mathbb{Z}_q$ and $\mathcal{M}_{\text{com}} = \{x \in \mathbb{Z} \mid |x| \leq R\}$ is: The former can have additive homomorphism (over $\mathbb{Z}_q$), but only binds to a representative of $x \in \mathbb{Z}_q$, not to an integer. The latter binds to a (bounded) integer, but has limited homomorphism (over $\mathbb{Z}$).

3.3 Zero-Knowledge Proofs

We define zero-knowledge with setup GenCRS, which generates a common reference string (CRS) $\mathsf{crs} \leftarrow \mathsf{GenCRS}(\mathsf{pp})$. In this work, we only require an unstructured CRS [15]. Let R be a NP-relation over a set X defining a (pp-dependent) NP-language $\mathscr{L} = \{x \in X \mid \exists w : \mathsf{R}(\mathsf{pp}, x, w) = 1\}$. For simplicity, we suppress the dependency on pp when it is clear. A zero-knowledge proof system for $\mathscr{L}$ is a protocol between a prover P and verifier V. We write $tr \leftarrow \langle \mathsf{P}(s), \mathsf{V}(t) \rangle$ for the transcript of an interaction where P (resp. V) has input s (resp. t) and *implicit inputs* $1^\lambda, \mathsf{pp}, \mathsf{crs}$. We write $b = \langle \mathsf{P}(s), \mathsf{V}(t) \rangle$ for the verifier's verdict b. A proof system is *public coin* if the verifier's messages are uniformly random and independent of the prover's messages, and the verifier outputs $b = \mathsf{Verify}(x, tr)$ for a PPT algorithm Verify.

Due to rejection sampling, our schemes have non-negligible correctness error.

Definition 5 (Correctness). A proof system $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ for $\mathscr{L}$ has *correctness error* γ_{err}, or is γ_{err}-correct, if for every adversary $\mathscr{A}$

$$\Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{GenPP}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{GenCRS}(\mathsf{pp}); \\ (x, w) \leftarrow \mathscr{A}(\mathsf{pp}, \mathsf{crs}) \colon \ \langle \mathsf{P}(x, w), \mathsf{V}(x) \rangle = 1 \end{array}\right] \geq 1 - \gamma_{\mathsf{err}}(\lambda)$$

We call $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ *correct* if $\gamma_{\mathsf{err}} = \text{negl}$.

To separate (statistical) simulation and knowledge errors from hardness assumptions as much as possible, we define zero-knowledge and knowledge extraction by means of adversary advantages.

Definition 6 (HVZK). A simulator Sim for a public coin proof system $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ for $\mathscr{L}$ is a PPT algorithm with input a statement x for which $(\mathsf{pp}, x, w) \in \mathsf{R}$ and implicit inputs $1^\lambda, \mathsf{pp}, \mathsf{crs}$, and output a transcript tr. Let $\mathscr{A}$ be a stateful algorithm and let

$$\mathsf{Real}_{\mathscr{A}}(\lambda) = \Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{GenPP}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{GenCRS}(\mathsf{pp}); (x, w) \leftarrow \mathscr{A}(\mathsf{pp}, \mathsf{crs}); \\ tr \leftarrow \langle \mathsf{P}(x, w), \mathsf{V}(x) \rangle; b \leftarrow \mathscr{A}(tr) \colon \ b \wedge \mathsf{R}(x, w) = 1 \end{array}\right]$$

$$\mathsf{Ideal}_{\mathscr{A}}(\lambda) = \Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{GenPP}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{GenCRS}(\mathsf{pp}); (x, w) \leftarrow \mathscr{A}(\mathsf{pp}, \mathsf{crs}); \\ tr \leftarrow \mathsf{Sim}(x); b \leftarrow \mathscr{A}(tr) \colon \ b \wedge \mathsf{R}(x, w) = 1 \end{array}\right]$$

Define the advantage of $\mathscr{A}$ by $\mathsf{Adv}^{\text{hvzk}}_{\mathscr{A},\mathsf{P},\mathsf{V}}(\lambda) = \mathsf{Real}_{\mathscr{A}}(\lambda) - \mathsf{Ideal}_{\mathscr{A}}(\lambda)$. Then Sim (and by extension $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$) is *honest verifier zero-knowledge* with *simulation error* $\sigma_{\mathsf{err}} = \sigma_{\mathsf{err}}(\lambda)$, if for all PPT $\mathscr{A}$ we have $\mathsf{Adv}^{\text{hvzk}}_{\mathscr{A},\mathsf{P},\mathsf{V}} \leq \sigma_{\mathsf{err}} + \text{negl}$.

Definition 7 (Knowledge error). Let $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ be a public coin proof system for $\mathscr{L}$. Let Ext be an *expected* polynomial time oracle algorithm (with oracle steps counted as one step) with implicit inputs $1^\lambda, \mathsf{pp}, \mathsf{crs}$. Let $\mathscr{A}$ be a (probabilistic) and P^* be a deterministic algorithm.

$$\mathsf{Real}_{\mathscr{A}}(\lambda) = \Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{GenPP}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{GenCRS}(\mathsf{pp}); (x, s) \leftarrow \mathscr{A}(\mathsf{pp}, \mathsf{crs}); \\ tr \leftarrow \langle \mathsf{P}^*(x, s), \mathsf{V}(x) \rangle \colon \ \mathsf{Verify}(x, tr) = 1 \end{array}\right]$$

$$\mathsf{Ideal}_{\mathscr{A}}(\lambda) = \Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{GenPP}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{GenCRS}(\mathsf{pp}); (x, s) \leftarrow \mathscr{A}(\mathsf{pp}, \mathsf{crs}); \\ (tr, w) \leftarrow \mathsf{Ext}^{\mathsf{P}^*(x,s)} \colon \ \mathsf{Verify}(x, tr) = 1 \wedge \mathsf{R}(x, w) = 1 \end{array}\right]$$

[15] Note that the distinction between structured and unstructured random strings is crucial in real-world applications: the former unavoidably requires either a trusted third party, or a secure distributed setup. However, the latter can be instantiated in the real-world using standard heuristic "nothing-up-my-sleeve" methods.

W.l.o.g. Ext let $w = \bot$ if $\mathsf{Verify}(x, tr) \neq 1$. The advantage of $(\mathscr{A}, \mathsf{P}^*)$ is $\mathsf{Adv}^{\mathrm{ke}}_{\mathscr{A},\mathsf{P}^*,\mathsf{V}}(\lambda) = \mathsf{Real}_{\mathscr{A}}(\lambda) - \mathsf{Ideal}_{\mathscr{A}}(\lambda)$. A proof system has knowledge error κ_{err}, if for any PPT $\mathscr{A}$, P^* we have $\mathsf{Adv}^{\mathrm{ke}}_{\mathscr{A},\mathsf{P}^*,\mathsf{V}} \leq \kappa_{\mathsf{err}} + \mathrm{negl}$.

Our definition of knowledge error is closely related to witness extended emulation [Lin03; GI08], which also requires that an extractor produces convincing transcripts. This property is trivial to achieve in our setting, but interferes with our definition of knowledge error. All of our proof systems are Σ-protocols.

Definition 8. A *Σ-protocol* Σ for relation R is an interactive three-move protocol consisting of four PPT algorithms $(\Sigma.\mathsf{Init}, \Sigma.\mathsf{Chall}, \Sigma.\mathsf{Resp}, \Sigma.\mathsf{Verify})$ between prover P holding a witness w for the statement $x \in \mathscr{L}$ and verifier V such that:

- $\Sigma.\mathsf{Init}(1^\lambda, w, x) \to (\alpha, \mathsf{st})$: On input of statement and witness (x, w) with $\mathsf{R}(x, w) = 1$, outputs a first message α and a state st.
- $\Sigma.\mathsf{Chall}(1^\lambda) \to \gamma$: Draw challenge γ uniformly from the set of challenges $[0, C]$.
- $\Sigma.\mathsf{Resp}(\mathsf{st}, \gamma) \to \omega$: On input of previous state st and challenge γ, outputs a response ω.
- $\Sigma.\mathsf{Verify}(x, \alpha, \gamma, \omega) \to b$: On input statement x and transcript α, γ, ω, accepts $(b = 1)$ or rejects $(b = 0)$.

Moreover, Σ must satisfy *correctness* and *HVZK*. As usual, the algorithms have implicit inputs $1^\lambda, \mathsf{pp}, \mathsf{crs}$.

The simulators for our Σ-protocols actually show *special HVZK*, that is, they work given any (adversarial) challenge γ. Letting Sim pick $\gamma \xleftarrow{\$} [0, C]$ yields standard HVZK. To prove knowledge extraction, we rely on k-special soundness.

Definition 9 (k-special soundness). A k-special soundness extractor Ext is a PPT algorithm which takes as input a set of k accepting transcripts $\Gamma = \{(\alpha, \gamma_i, \omega_i) \mid \Sigma.\mathsf{Verify}(x, \alpha, \gamma_i, \omega_i) = 1\}_{i=1..k}$ with fixed α and pair-wise distinct challenges γ_i, and outputs a valid witness $w \leftarrow \mathsf{Ext}(\Gamma)$, i.e. $\mathsf{R}(w, x) = 1$.

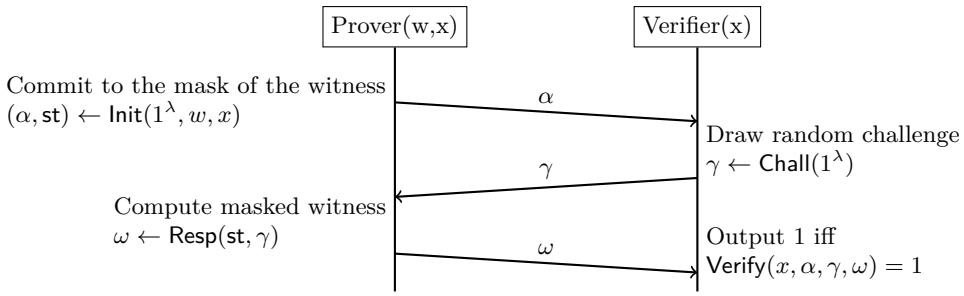
In security proofs, k transcripts will either yield a witness or break an assumption. Formally, we consider the language $\mathscr{L} \vee \mathscr{L}_{\mathrm{hard}}$ instead of $\mathscr{L}$. We recall how to obtain k transcripts for a special-soundness extractor as in definition 9.

Remark 1 (Getting k transcripts). If the adversary's success probability is at least $k/((C + 1))$, this is easily solved in *expected* polynomial time. Just rewind and try fresh challenge. In expected constant rewinds, a second accepting challenge γ will be found. The probability that γ was not encountered before is at least $1/k$. Since $k = \mathsf{poly}$ in expected polynomial time, k distinct transcript are found. Thus, we can assume w.l.o.g. that we have k transcripts as in definition 9.

Instead of relying on *expected* polynomial time, one can reduce to strict PPT by truncation while retaining inversely polynomial advantage, but this incurs a hard to quantify loss. Alternatively, by standard arguments, e.g. the forking lemma [PS00], k rewinds yield k accepting transcripts with probability at least $(\frac{\varepsilon}{2})^k$ and these are as in definition 9 with probability at least $\frac{(k-1)!}{(\varepsilon \cdot (C+1))^k}$.

Definition 10 (Fiat–Shamir Transformation). The Fiat–Shamir transformation applied to a Σ-protocol replaces the verifier's random challenge by a hash, resulting in a non-interactive proof system. Concretely, the prover computes $(\alpha, \mathsf{st}) \leftarrow \Sigma.\mathsf{Init}(w, x)$, then $\gamma = \mathsf{H}(x, \alpha)$ and $\omega \leftarrow \mathsf{Resp}(\mathsf{st}, \gamma)$. This results in a non-interactive proof $\pi = (\alpha, w)$, since a verifier can check $\Sigma.\mathsf{Verify}(x, \alpha, \gamma, \omega)$ by recomputing $\gamma = \mathsf{H}(x, \alpha)$. Our Σ-protocols satisfy the property, that given x, γ, ω, the unique accepting α can be computed efficiently. Thus, an alternative, often shorter, non-interactive proof $\pi = (\gamma, \omega)$ is possible. Now, the verifier computes α, and checks $\gamma = \mathsf{H}(x, \alpha)$ and $\Sigma.\mathsf{Verify}(x, \alpha, \gamma, \omega)$. Security can be proven if H is modelled as a random oracle (and C is superpolynomial). The technique can be extended to work with random aborts of the prover [Lyu09].

Fig. 1. The execution of a Σ-protocol between a prover and a verifier.



Range Proofs. A range proof is essentially a zero-knowledge proof that guarantees that a committed value x resides inside a specified interval $[a, b]$. We can show so by setting $y = (b - x)(x - a)$, computing the commitment to y homomorphically from the commitment to x and the constants a, b, and showing that $y \geq 0$ in a zero-knowledge manner. The following lemma yields a strategy to show that committed integers are non-negative.

Lemma 1 (Decomposition into 3 Squares [RS86; Gro05]). *Let $y \in \mathbb{Z}$ be an integer. It holds that*

$$y \geq 0 \iff \exists \{x_i\}_{i=1..3} : 4y + 1 = \sum_{i=1..3} x_i^2$$

Further, the integers x_i can be efficiently computed. In [PS19], the runtime of finding the decomposition was improved to $\mathcal{O}(\log^2(y)/\log\log(y))$ multiplications.

3.4 Tools in the DLOG setting

Hardness Assumptions. First, we establish the hardness assumptions that our scheme in the DLOG setting is based on (see section 5). To avoid trusted setup, we assume a deterministic family $\mathbb{G} = \mathbb{G}_\lambda$ of cyclic groups with generator g_λ and known order q_λ, generated by a group generator $(\mathbb{G}_\lambda, g_\lambda, q_\lambda) = \mathsf{GenGrp}(1^\lambda)$. For notational simplicity, we leave GenGrp implicit in the rest of the work. Throughout we assume that $\mathbb{G}$ has *invertible sampling for random group elements*. This is necessary in the security proofs for "programming" the CRS (despite transparent setup).

Definition 11 (S-Bounded DLSE and SEI Assumption). Consider a group $\mathbb{G}$ of order q with generator g. Let $S < q$. The S-bounded DLSE assumption holds if for all PPT $\mathscr{A}$ there is a negligible negl such that

$$\Pr\left[z \xleftarrow{\$} \{0..S-1\}, z' \leftarrow \mathscr{A}(g^z) \colon\ z = z'\right] \leq \mathrm{negl}(\lambda)$$

The S-bounded short exponent indistinguishability (SEI) assumption holds if for all PPT $\mathscr{A}$ there is a negligible negl such that

$$\left|\Pr\left[z \xleftarrow{\$} \{0..S-1\} \ :\ \mathscr{A}(g^z) = 1\right] - \Pr\left[z \xleftarrow{\$} \mathbb{Z}_{\mathsf{ord}} \ :\ \mathscr{A}(g^z) = 1\right]\right| \leq \mathrm{negl}(\lambda)$$

Throughout this work, we generally set $S = 2^{2\lambda}$. Note that DLOG assumption is equivalent to the q-bounded DLSE assumption.

Tools. Now, we introduce some lemmas and a commitment scheme that we later on utilize for constructing the bounded integer commitment and range proof.

Lemma 2 ([KK04]). *Let $\mathbb{G}$ be a group of prime order q with generator $g \in \mathbb{G}$. For $S < q/2$, the S-bounded DLSE and SEI assumptions are equivalent.*

We consider a Pedersen commitment scheme [Ped92] with smaller openings in exchange for a computational (instead of statistical) hiding property.

Definition 12 (Pedersen Commitments with Short Openings.). Let $\mathbb{G}$ be a group of prime order q. The scheme Ped consists of a 3-tuple of PPT algorithms ($\mathsf{Ped.Setup}$, $\mathsf{Ped.Commit}$, $\mathsf{Ped.Verify}$) such that

- $\mathsf{Ped.Setup}(1^\lambda)$: samples $g, h \xleftarrow{\$} \mathbb{G}$ and outputs public parameters $\mathsf{pp} = (g, h)$,
- $\mathsf{Ped.Commit}_{\mathsf{pp}}(x)$: samples $d \xleftarrow{\$} [0, 2^{2\lambda}]$ for $x \in \mathbb{Z}_q$, sets $c = g^x h^d$ and outputs the pair (c, d),
- $\mathsf{Ped.Verify}_{\mathsf{pp}}(c, x, d)$: outputs 1 iff $c = g^x h^d$.

Using $d \xleftarrow{\$} [0, 2^{2\lambda}]$ instead of $d \xleftarrow{\$} [0, q-1]$ (as in [Ped92]) still achieves computational hiding: Under SEI (or equivalently DLSE), we can replace the short random exponent d in h^d with a full random $d \xleftarrow{\$} [0..q-1]$ in a hybrid game. Now $g^x h^d$ is uniformly distributed, independent of x.

3.5 Tools in the Lattice setting

Hardness Assumptions. Now, we state the required hardness assumptions for section 6. Note that we generally instantiate the distribution χ of the LWE assumption with a discrete Gaussian distribution defined further below.

$\mathsf{LWE}_{n,m,q,\chi}$ (Normal Form). Given a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{(m-n)\times n}$ and a vector $\vec{b} \in \mathbb{Z}_q^{m-n}$ generated according to either of the following cases:

1. $\vec{b} = \boldsymbol{A} \cdot \vec{s} + \vec{e}$ where $\vec{s} \xleftarrow{\$} \chi^{m-n}$ and $\vec{e} \xleftarrow{\$} \chi^n$
2. $\vec{b} \xleftarrow{\$} \mathbb{Z}_q^{m-n}$

distinguish which is the case.

$\mathsf{SIS}_{n,m,q,\beta}$ (Normal Form). Given a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times(m-n)}$, find a non-zero $\vec{z} \in \mathbb{Z}^m$ s.t. $\|z\| \leq \beta$ and $\begin{bmatrix}\boldsymbol{I}_n & \boldsymbol{A}\end{bmatrix} \cdot \vec{z} = \vec{0}$.

Tools. Gaussian distributions and related lemmata introduced in section 3.7, such as rejection sampling, are standard tools in the lattice setting. Now, we state the commitment scheme from which we construct a bounded integer commitment scheme.

Definition 13 (Commitments with Relaxed Opening in the Lattice Setting.). The following commitment scheme was defined in [Bau+18] and adapted to the standard LWE setting in [Yan+19]. Let $q \in \mathbb{N}$ be a large enough power of a prime and $n \in \mathbb{N}$. Let $l_1, l_2, C \in \mathsf{poly}(\lambda)$ and $\sigma \geq \sqrt{2l_2/\pi}$ be positive integers. We define $l := l_1 + l_2 + n$. Let $T \in \mathbb{N}$ such that $T \geq \sigma\sqrt{l}$. The commitment scheme with $\mathcal{M}_{\mathrm{com}} = \mathbb{Z}_q^n$ is defined as follows:

- $\mathsf{Lat.Setup}(1^\lambda)$: for $\boldsymbol{A}_1 \xleftarrow{\$} \mathbb{Z}_q^{l_1\times(l_2+n)}$, $\boldsymbol{A}_2 \xleftarrow{\$} \mathbb{Z}_q^{n\times l_2}$, sets
$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_{l_1} & \boldsymbol{A}_1 \\ \boldsymbol{0}^{n\times l_1} & \boldsymbol{I}_n \ \boldsymbol{A}_2 \end{bmatrix}$$
and outputs $\mathsf{pp} = \boldsymbol{A}$.
- $\mathsf{Lat.Commit}_{\mathsf{pp}}(x)$: for $\vec{x} \in \mathcal{M}_{\mathrm{com}}$, samples $\vec{r} \xleftarrow{\$} D_\sigma^l$, sets $\vec{c} = \boldsymbol{A} \cdot \vec{r} + (\vec{0} \,\|\, \vec{x})$ and outputs the pair $(\vec{c}, (\vec{r}, 1))$.
- $\mathsf{Lat.Verify}_{\mathsf{pp}}(\vec{c}, \vec{x}, (\vec{r}, f))$: for $\vec{x} \in \mathcal{M}_{\mathrm{com}}$ with the opening $(\vec{r}, f) \in \mathbb{Z}^{l_1+n+l_2} \times \mathbb{Z}$ checks if
$$\begin{aligned} f \cdot \vec{c} &= \boldsymbol{A} \cdot \vec{r} + f \cdot (\vec{0} \,\|\, \vec{x}) \\ \|\vec{r}\| &\leq T \\ |f| &\leq 2C \end{aligned}$$

Note that a commitment has bit-size $(l_1 + n) \cdot \log(q)$. In the following, we extract the proof of security of Lat of [Yan+19] from the proof of security of their zero-knowledge scheme.

Lemma 3. *The commitment scheme* Lat *is correct.*

Proof. We have $f = 1 \leq 2C$, by design $\vec{c} = \vec{A} \cdot \vec{r} + (\vec{0} \,\|\, \vec{x})$ and using lemma 10 also with overwhelming probability $\|\vec{r}\| \leq \sigma\sqrt{l_1 + l_2 + n} \leq T$.

Lemma 4. *The commitment scheme* Lat *is binding under the* $\mathsf{SIS}_{l_1,(l_2+n+l_1),q,\beta}$ *assumption for any* $\beta \geq 4CT$.

Proof. Given a SIS instance $\boldsymbol{A}_1 \in \mathbb{Z}_q^{l_1 \times (l_2+n)}$, sample $\boldsymbol{A}_2 \xleftarrow{\$} \mathbb{Z}_q^{n \times l_2}$ and set $\mathsf{pp} = \boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_{l_1} & \boldsymbol{A}_1 \\ \boldsymbol{0}^{n \times l_1} & \boldsymbol{I}_n \; \boldsymbol{A}_2 \end{bmatrix}$. Assume an adversary given pp can output $(\vec{c}, (\vec{r}_0, f_0), (\vec{r}_1, f_1), \vec{x}_0, \vec{x}_1)$ such that $\vec{m}_0 \neq \vec{m}_1$ and $\mathsf{Lat.Verify}_{\mathsf{pp}}(\vec{c}, \vec{x}_0, (\vec{r}_0, f_0)) = \mathsf{Lat.Verify}_{\mathsf{pp}}(\vec{c}, \vec{x}_1, (\vec{r}_1, f_1)) = 1$, so:

$$
\begin{aligned}
& f_0 \cdot \vec{c} = \boldsymbol{A} \cdot \vec{r}_0 + f_0 \cdot (\vec{0} \,\|\, \vec{x}_0) \;\wedge \\
& f_1 \cdot \vec{c} = \boldsymbol{A} \cdot \vec{r}_1 + f_1 \cdot (\vec{0} \,\|\, \vec{x}_1) \\
\implies & f_1 f_0 \cdot \vec{c} = f_1 \boldsymbol{A} \cdot \vec{r}_0 + f_1 f_0 \cdot (\vec{0} \,\|\, \vec{x}_0) \;\wedge \\
& f_0 f_1 \cdot \vec{c} = f_0 \boldsymbol{A} \cdot \vec{r}_1 + f_0 f_1 \cdot (\vec{0} \,\|\, \vec{x}_1) \\
\implies & f_0 \boldsymbol{A} \cdot \vec{r}_1 + f_0 f_1 \cdot (\vec{0} \,\|\, \vec{x}_1) = f_1 \boldsymbol{A} \cdot \vec{r}_0 + f_1 f_0 \cdot (\vec{0} \,\|\, \vec{x}_0) \\
\implies & \boldsymbol{A} \cdot (f_0 \vec{r}_1 - f_1 \vec{r}_0) + f_0 f_1 \cdot (\vec{0} \,\|\, \vec{x}_1 - \vec{x}_0) = \vec{0} \\
\implies & \begin{bmatrix} \boldsymbol{I}_{l_1} & \boldsymbol{A}_1 \end{bmatrix} \cdot (f_0 \vec{r}_1 - f_1 \vec{r}_0) = \vec{0}
\end{aligned}
$$

So $\vec{z} = (f_0 \vec{r}_1 - f_1 \vec{r}_0)$ is a solution to the SIS instance because additionally it holds that:

$$
\begin{aligned}
\|f_0 \vec{r}_1 - f_1 \vec{r}_0\| &\leq \|f_0 \vec{r}_1\| + \|f_1 \vec{r}_0\| \\
&\leq 2 \cdot (2CT) \\
&\leq \beta
\end{aligned}
$$

Lemma 5. *The commitment scheme* Lat *is hiding under the* $\mathsf{LWE}_{l_2,l_1+n+l_2,q,D_\sigma}$ *assumption.*

Proof. On input of LWE instance $(\overline{\boldsymbol{A}}, \bar{\vec{b}}) \in \mathbb{Z}_q^{(l_1+n) \times l_2} \times \mathbb{Z}_q^{l_1+n}$, sample matrix $\boldsymbol{R} \xleftarrow{\$} \mathbb{Z}_q^{l_1 \times n}$. Set

$$
\boldsymbol{B} = \begin{bmatrix} \boldsymbol{I}_{l_1} & \boldsymbol{R}\,\overline{\boldsymbol{A}}_U + \boldsymbol{R}\overline{\boldsymbol{A}}_L \\ \vec{0}^{n \times l_1} & \boldsymbol{I}_n \qquad \overline{\boldsymbol{A}}_L \end{bmatrix}, \vec{c} = \begin{bmatrix} \bar{\vec{b}}_U + \boldsymbol{R} \cdot \bar{\vec{b}}_L \\ \bar{\vec{b}}_L \end{bmatrix}
$$

for $\overline{\boldsymbol{A}} = \begin{bmatrix} \overline{\boldsymbol{A}}_U \\ \overline{\boldsymbol{A}}_L \end{bmatrix}$ and $\bar{\vec{b}} = \begin{bmatrix} \bar{\vec{b}}_U \\ \bar{\vec{b}}_L \end{bmatrix}$. Since $\overline{\boldsymbol{A}}$ is random, public parameter $\mathsf{pp} = \boldsymbol{B}$ is distributed as in $\mathsf{Lat.Setup}$ and in case of $\bar{\vec{b}} = \vec{A} \cdot \vec{s} + \vec{e}$ for $\vec{s} \xleftarrow{\$} D_\sigma^{l_2}$ and $\vec{e} \xleftarrow{\$} D_\sigma^{l_1+n}$, $\vec{c}$ is distributed as a commitment to $\vec{x}$ and otherwise a random element of $\mathbb{Z}_q^{l_1+n}$.

In summary, the security of the commitment scheme Lat relies on the $\mathsf{LWE}_{l_2,l_1+n+l_2,q,D_\sigma}$ assumption and the $\mathsf{SIS}_{l_1,(l_2+n+l_1),q,\beta}$ assumption with $\beta \geq 4CT$.

3.6 Tools in the Class Group Setting

We recap the definition of class groups and present the setting considered in this work.

Class Groups. A class group $\mathbb{G} = \mathsf{Cl}(\Delta)$ of an imaginary quadratic order is the quotient group of fractional ideals by principal ideals of the group $\mathbb{Q}(\sqrt{\Delta})$ with ideal multiplication. It is defined by its discriminant Δ which must satisfy $\Delta \equiv 1 \mod 4$ and $-\Delta$ must be a prime. Note that the Δ can be generated from public coins for a given security parameters λ [BFS20]. However, unlike for typical prime order groups, there is no (commonly known) sampling algorithm for uniform group elements which is also inverse sampleable. We can compute bounds of the order of the group $L, U \in \mathsf{poly}(\lambda)$ such that $2^L \leq \mathsf{ord}(\mathbb{G}) \leq 2^U$; however, we will not need L. We denote by $(g, \rho) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{G})$ a sampling algorithm for drawing uniformly random elements g from $\mathbb{G}$ with random coins ρ.

Note that there exists an efficient algorithm for computing square roots of arbitrary elements [BS96], and we make explicit use of this fact. Thus class groups can be used to commit to dyadic rationals $\frac{a}{2^k}$ for $a, k \in \mathbb{Z}$. Applying the encoding technique from section 4, we can map dyadic rationals to unique integers while retaining homomorphic properties. This gives rise to an unbounded integer commitment scheme without trusted setup described in section 7.

Lemma 6 (Random Group Elements). *For given $g \in \mathsf{Cl}(\Delta)$, the statistical distance of the distributions $\mathcal{U}\langle g\rangle$ and $\{g^x \mid x \xleftarrow{\$} [0, 2^U \cdot L]\}$ is at most L^{-1}, where 2^U is an upper bound of $\mathsf{ord}(\mathbb{G})$ and $L \geq 1$ is an arbitrary integer.*

Proof. This follows similarly as in lemma 8 and we omit the details.

Assumptions. Due to the lack of invertible sampling of random group elements in class groups, we cannot "program" a transparent CRS in our reductions. Therefore, we base the security of our scheme upon the DXDH assumption of Abram et al. [Abr+22] (who also notified us of this problem[16]) which makes the sampling coins an explicit output.

We introduce the DXDH assumption [Abr+22] (tailored to class groups).

Definition 14 (DXDH). The decisional cross-group Diffie-Hellman assumption (DXDH) holds for a given group $\mathbb{G}$ and exponent upper bound S if for any PPT adversary $\mathscr{A}$ it holds that

$$\Pr\left[\begin{array}{l}(g,\rho_g) \leftarrow \mathsf{Sample}(1^\lambda,\mathbb{G}), (h,\rho_h) \leftarrow \mathsf{Sample}(1^\lambda,\mathbb{G}) \\ r \xleftarrow{\$} [0,S], s \xleftarrow{\$} [0,S] \\ d_0 \leftarrow h^r, d_1 \leftarrow g^s \\ b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathscr{A}(\mathbb{G}, g, \rho_g, h, \rho_h, g^r, d_b)\end{array} : b = b'\right] \leq \frac{1}{2} + \mu(\lambda),$$

The following assumptions are discussed in [BFS20] and are believed to hold in class groups.[17] Again, we denote explicitly the public coins of sampled group elements.

Definition 15 (ORD). The order (ORD) assumption holds for a given group $\mathbb{G}$ if for any PPT adversary $\mathscr{A}$ it holds that

$$\Pr\left[\begin{array}{l}(w,\alpha) \leftarrow \mathscr{A}(\mathbb{G}) \\ w \in \mathbb{G} \setminus \{1\}, \\ 0 \neq |\alpha| < 2^{\mathsf{poly}(\lambda)}\end{array} : w^\alpha = 1\right] \leq \mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$.

To account for the lack of invertible sampling, we have adapted the r-fROOT assumption to explicitly pass the adversary the sampling coins.

Definition 16 (r-fROOT). The r-fractional root (r-fROOT) assumption holds for group $\mathbb{G}$ if for any PPT adversary $\mathscr{A}$ it holds that

$$\Pr\left[\begin{array}{l}(g,\rho) \leftarrow \mathsf{Sample}(1^\lambda,\mathbb{G}) \\ (\alpha,\beta,u) \leftarrow \mathscr{A}(\mathbb{G}, g, \rho) \\ 0 \neq |\alpha| < 2^{\mathsf{poly}(\lambda)} \in \mathbb{Z} \\ |\beta| < 2^{\mathsf{poly}(\lambda)} \in \mathbb{Z}\end{array} : u \in \mathbb{G} \wedge u^\beta = g^\alpha \wedge \frac{\beta}{\gcd(\alpha,\beta)} \neq r^k \text{ for } k \in \mathbb{N}\right] \leq \mu(\lambda)$$

Note that the 2-fROOT assumption is believed to hold in the setting of class groups of imaginary quadratic orders.

[16] In the original version of this range proof, we used Pedersen commitment instead of ElGamal commitments. These lead to problems due to a lack of invertible sampling. See appendix B for a discussion of changes.

[17] Moreover, we adapted the version of [BFS20] to exclude zero solutions. This refers to excluding that $\alpha = 0$ in the following assumptions. This condition is not checked in the definitions of [BFS20] but assumed to be true in the reductions between common RSA group assumptions, namely the adaptive root assumption and the strong RSA assumption, and the order and fractional root assumption. Further, the problems in the given assumptions would be trivial if $\alpha = 0$ was accepted as solution.

Lemma 7 (Forcing Zero Exponents). *For any PPT adversary $\mathscr{A}$ it holds that*

$$\Pr\left[\begin{array}{l}(g,\rho_g)\leftarrow \mathsf{Sample}(1^\lambda,\mathbb{G}),(h,\rho_h)\leftarrow \mathsf{Sample}(1^\lambda,\mathbb{G})\\(\alpha,\beta)\leftarrow \mathscr{A}(\mathbb{G},g,\rho_g,h,\rho_h) \qquad : g^\alpha = h^\beta \wedge \alpha\neq 0\\|\alpha|,|\beta|<2^{\mathsf{poly}(\lambda)}\end{array}\right]\leq \mu(\lambda)$$

under the DXDH *assumption with exponent upper bound $S=2^{U+\lambda}$ and the* ORD *assumption.*

Proof. Let $\mathscr{A}$ be an adversary on the above game. We construct an adversary $\mathscr{B}$ for DXDH. First, $\mathscr{B}$ receives $\mathbb{G},g,\rho_g,h,\rho_h,R,D_b$, where $R=g^r$ and D_b is either h^r or g^s for random $r,s\in[0,S]$. Now, $\mathscr{B}$ invokes $\mathscr{A}$ on input $\mathbb{G},g,\rho_g,h,\rho_h$ and receives α,β. Next, $\mathscr{B}$ verifies that indeed $g^\alpha=h^\beta\wedge\alpha\neq 0$, and if not, it outputs a random bit $b'\xleftarrow{\$}\{0,1\}$. If otherwise $\mathscr{A}$'s output is a correct solution, $\mathscr{B}$ checks whether $R^\alpha=D_b^\beta$. If yes, $\mathscr{B}$ outputs 0, else 1.

We now analyze the success probability of $\mathscr{B}$. First, note that $\mathscr{B}$ outputs a random bit if $\mathscr{A}$ is not successful. Otherwise, if $\mathscr{A}$ is successful, it holds that $g^\alpha=h^\beta$ and $\alpha\neq 0$.

If $b=0$, then $D_b=h^r$. In that case, $R^\alpha=(g^\alpha)^r=(h^\beta)^r=D_b^r$ and $\mathscr{B}$ outputs 0.

If $b=1$, then $D_b=g^s$. Here, $R^\alpha-D_b^\beta=g^{r\alpha-s\beta}$. Here, $g^{r\alpha-s\beta}\neq 1$ under ORD with overwhelming probability and thus, $\mathscr{B}$ outputs 1 as $R^\alpha\neq D_b^\beta$. (It is easy to see, that $r\alpha-s\beta=0$ (over $\mathbb{Z}$) happens with negligible probability, since informationtheoretically only $r_0\coloneqq r \mod \mathsf{ord}(g)$ is known to $\mathscr{A}$, but $r=r_0+k\cdot\mathsf{ord}(g)$, and k is almost uniform in $\{0,\dots,\lfloor 2^{U+\lambda}/\mathsf{ord}(g)\rfloor\}$. So even if s is known, it is hard to predict α,β s.t. $r\alpha=s\beta$.)

In conclusion, we can either construct an adversary for DXDH with advantage $\varepsilon/2$ or an adversary on ORD with advantage $(1-\mathrm{negl})\cdot\varepsilon/2$, where ε is the advantage of $\mathscr{A}$.

3.7 Tools for Zero-Knowledge

As a technical tool for achieving zero knowledge, our protocols use additive masking of the witness. We recall the tools for masking here.

Lemma 8 (Masking with the Security Parameter). *For any $C,B,L\in\mathbb{N}$ and fixed $x\in[-B,B],\gamma\in[-C,C]$, the distributions $U=\mathcal{U}[0,BCL]$ and $V=\{m+\gamma\cdot x\mid m\xleftarrow{\$}[0,BCL]\}$ have statistical distance at most $1/L$.*

Rejection sampling and Gaussian noise allow to use smaller masks.

Definition 17 (Discrete Gaussian Distributions, [Yan+19]). The continuous Gaussian distribution over $\mathbb{R}^m$ centered around $\vec{v}\in\mathbb{R}^m$ with standard deviation σ is defined by the density function $\rho^m_{\vec{v},\sigma}(\vec{x})=(\frac{1}{\sqrt{2\pi\sigma^2}})^m e^{\frac{-\|\mathbf{x}-\mathbf{v}\|_2^2}{2\sigma^2}}$. The discrete Gaussian distribution over $\mathbb{Z}^m$ centered around $\vec{v}\in\mathbb{Z}^m$ with standard deviation σ is defined as $D^m_{\vec{v},\sigma}(\vec{x})=\rho^m_{\vec{v},\sigma}(\vec{x})/\rho^m_\sigma(\mathbb{Z}^m)$, where $\rho^m_\sigma(\mathbb{Z}^m)=\sum_{x\in\mathbb{Z}^m}\rho^m_\sigma(x)$. We write $D^m_\sigma(\vec{x})=D^m_{\vec{0},\sigma}(\vec{x})$ for short.

Lemma 9 (Relationship between norms). *For $v\in\mathbb{R}^m$, the inequalities of norms, $\|v\|_\infty\leq\|v\|_1\leq\sqrt{N}\|v\|_2\leq N\|v\|_\infty$, are well known.*

Lemma 10 (Lemma 4.4, [Lyu12]).

- *For any $k>0$ it holds that $\Pr[|z|>k\sigma\mid z\xleftarrow{\$}D_\sigma]\leq 2e^{\frac{-k^2}{2}}$.*
- *For any $k>1$ it holds that $\Pr[\|\vec{z}\|_2>k\sigma\sqrt{m}\mid\vec{z}\xleftarrow{\$}D^m_\sigma]<k^m e^{\frac{m}{2}(1-k^2)}$.*

Lemma 11 (Theorem 4.6, [Lyu12]). *Let V be a subset of $\mathbb{Z}^m$ in which all elements have $\|\cdot\|_2$ norms less than T, $\sigma\in\mathbb{R}$ such that $\sigma=\omega(T\sqrt{\log m})$ and $h:V\mapsto\mathbb{R}$ a probability distribution. Define algorithms $\mathscr{T}$ (resp. $\mathscr{S}$) as follows:*

1. *$\vec{v}\xleftarrow{\$}h$*
2. *$\vec{t}\xleftarrow{\$}D^m_{\vec{v},\sigma}$ (resp. $\vec{t}\xleftarrow{\$}D^m_\sigma$)*
3. *output $(\vec{t},\vec{v})$ with probability $\min\left(\frac{D^m_\sigma(\vec{t})}{M\cdot D^m_{\vec{v},\sigma}(\vec{t})},1\right)$ (resp. with probability $1/M$)*

Then there exists a constant $M = O(1)$ such that the output distributions of $\mathscr{T}$ and $\mathscr{S}$ are within statistical distance $\frac{2^{-\omega(\log m)}}{M}$. Moreover, the probability that $\mathscr{T}$ outputs something is at least $\frac{1-2^{-\omega(\log m)}}{M}$.

Note that if $\sigma = \alpha T$ for some $\alpha > 0$, then $M = e^{13.3/\alpha + 1/(2\alpha^2)}$, the output of algorithm $\mathscr{T}$ is within statistical distance $2^{-128}/M$ of the output of $\mathscr{S}$ and the probability that $\mathscr{T}$ outputs something is at least $\frac{1-2^{-128}}{M}$ [Yan+19; Hof+17].

4 Integer Commitments from Rounding Fractions

In this section, we introduce bounded integer commitments and motivate the construction of range proofs based on these commitments.

4.1 Bounded Integer Commitment Scheme

We introduce a commitment scheme transformation that allows to commit to bounded integers. The core feature of this transformation is its proof-friendliness: standard Σ-protocols for proving knowledge of a square decomposition (or, more generally, any low-degree polynomial relation) with the original commitment (over a field $\mathbb{Z}_q$) can be re-interpreted (with minor adaptations) as Σ-protocols for proving knowledge of a square decomposition (resp. low-degree relation) over $\mathbb{Z}$ with respect to the transformed commitment scheme. In addition, the transformation preserves some homomorphic properties of the underlying scheme, which turns out to be crucial in the application to range proofs.

Definition 18 (The Transformation). Let com be a commitment scheme with message space $\mathsf{com}.\mathcal{M}_{\mathrm{com}} = \mathbb{Z}_q^n$ and opening space $\mathsf{com}.\mathcal{R}_{\mathrm{com}}$. We define the commitment scheme $\overline{\mathsf{com}}$ over parameters $U, C \in \mathbb{N}$ such that $U < \frac{q-1}{2}$ with

- $\overline{\mathsf{com}}.\mathcal{M}_{\mathrm{com}} = \{\vec{x} \in \mathbb{Z}^n \mid \|\vec{x}\|_\infty \leq U/C\}$
- $\overline{\mathsf{com}}.\mathcal{R}_{\mathrm{com}} = \{(d, \gamma, \vec{y}) \in \mathcal{R}_{\mathrm{com}} \times \mathbb{Z} \times \mathbb{Z}^n \mid \gamma \leq C, \|\vec{y}\|_\infty \leq U/C\}$

as follows:

- $\overline{\mathsf{com}}.\mathsf{Setup}(1^\lambda)$: outputs $\mathsf{pp} \leftarrow \mathsf{com}.\mathsf{Setup}(1^\lambda)$.
- $\overline{\mathsf{com}}.\mathsf{Commit}_{\mathsf{pp}}(\vec{x})$: computes $(c, r) \leftarrow \mathsf{com}.\mathsf{Commit}_{\mathsf{pp}}(\vec{x})$ and outputs $(c, (r, 1, \vec{x}))$.
- $\overline{\mathsf{com}}.\mathsf{Verify}_{\mathsf{pp}}(c, \vec{x}, (r, \gamma, \vec{y}))$: sets $\vec{z} = \vec{y} \cdot \gamma^{-1} \mod q$ and checks $\vec{x} = \lfloor \frac{\vec{y}}{\gamma} \rceil, |\gamma| \leq C, \gamma \neq 0, \|\vec{y}\|_\infty \leq U/C, \mathsf{com}.\mathsf{Verify}_{\mathsf{pp}}(c, \vec{z}, r) = 1$ as well as $\vec{x} = \lfloor \frac{\vec{y}}{\gamma} \rceil$, where division is performed in $\mathbb{Q}^n$.

Lemma 12. *The commitment scheme $\overline{\mathsf{com}}$ is correct, binding and hiding.*

The correctness and hiding properties follow directly from the security of com. The binding property can be argued similarly.

Let $\mathscr{A}$ be a PPT adversary breaking the binding property of $\overline{\mathsf{com}}$. We design a PPT adversary $\mathscr{B}$ that breaks the binding property of com with challenger $\mathcal{C}$.

On receiving pp from the challenger $\mathcal{C}$, $\mathscr{B}$ forwards pp to $\mathscr{A}$ and receives $(c, (d_0, \gamma_0, \vec{y}_0), ((d_1, \gamma_1, \vec{y}_1), \vec{x}_0, \vec{x}_1)$. $\mathscr{B}$ sets $\vec{z}_i = \vec{y}_i \cdot \gamma_i^{-1} \mod q$ and just forwards $(c, d_0, d_1, \vec{z_0}, \vec{z_i})$ to $\mathcal{C}$. If $\mathscr{A}$ is successful, both commitments verify correctly with respect to $\overline{\mathsf{com}}$ and $\vec{x}_0 \neq \vec{x}_1$. Thus by definition of $\overline{\mathsf{com}}.\mathsf{Verify}$, the verification check for the sent openings are valid with respect to the scheme com. Note that $\|\vec{y}_i\|_\infty \leq U/C, |\gamma_i| \leq C$ for $i \in [0, 1]$. So $\|\vec{y}_i \cdot \gamma_i\|_\infty \leq U \leq \frac{q-1}{2}$. Assume for the sake of contradiction that $\vec{z}_0 = \vec{z}_1$:

$$\begin{aligned} \vec{z}_0 = \vec{z}_1 &\implies \vec{y}_0 \cdot \gamma_1 = \vec{y}_1 \cdot \gamma_0 \mod q \implies \vec{y}_0 \cdot \gamma_1 = \vec{y}_1 \cdot \gamma_0 \text{ in } \mathbb{Q} \\ &\implies \frac{\vec{y}_0}{\gamma_0} = \frac{\vec{y}_1}{\gamma_1} \text{ in } \mathbb{Q} \implies \left\lfloor \frac{\vec{y}_0}{\gamma_0} \right\rceil = \left\lfloor \frac{\vec{y}_1}{\gamma_1} \right\rceil \text{ in } \mathbb{Q} \end{aligned}$$

This contradicts $\vec{x}_0 \neq \vec{x}_1$ and thus the advantage of $\mathscr{B}$ is the same as $\mathscr{A}$.

Arguing over the Integers. Now, we motivate how to perform proofs over the integers on the example Ped. Let $\overline{\mathsf{Ped}}$ be the scheme obtained by the above transformation applied to Ped. Let $C = 2^\lambda$ determine the challenge space, $S = 2^{2\lambda}$ determine the size of the randomness and $L = 2^\lambda$ be the masking overhead. Let $2^\lambda = C < U \in \mathbb{N}$ and let q be prime with $2U < q$. Let $\mathbb{G}$ be a group of order q. For clarity, we restate the scheme:

- $\overline{\mathsf{Ped}}.\mathsf{Setup}(1^\lambda)$: outputs $\mathsf{pp} = (g, h) \xleftarrow{\$} \mathbb{G}^2$.
- $\overline{\mathsf{Ped}}.\mathsf{Commit}(\mathsf{pp}, x)$: samples $r \xleftarrow{\$} [0, S]$ and outputs $(c = g^x h^r, (r, 1, x))$.
- $\overline{\mathsf{Ped}}.\mathsf{Verify}(\mathsf{pp}, c, x, (r, \gamma, y))$: checks $g^{y \cdot \gamma^{-1}} h^r = c$ as well as $x = \lfloor \frac{y}{\gamma} \rceil$, where the division is performed in $\mathbb{Q}$, $|\gamma| \leq C, \gamma \neq 0$ and $|y| \leq U/C$.

The most essential protocol is the proof of knowledge of an opening. We now establish an unoptimized version in order to gain a basic understanding of the underlying arguments. The relation we prove is

$$\mathsf{R} = \{(c, (x, (r, \gamma, y))) \mid \overline{\mathsf{Ped}}.\mathsf{Verify}(c, x, (r, \gamma, y)) = 1\}.$$

For the correctness property, we are only interested in honest openings, so $\gamma = 1, y = x$. The proof scheme follows the conventional strategy of blinding the witnesses (x, r) with a mask. We add a size check for the masked witness to ensure the shortness of the opening. Note that the message space of $\overline{\mathsf{Ped}}$ is $\{x \in \mathbb{Z} \mid x \leq U/C\}$ but we can only perform proofs for smaller x values because the commitments need to stay binding after the masking process. In more detail, we let $B \in \mathbb{N}$ such that $2BCL \leq U/C$ and we allow for messages $|x_i| \leq B$. The following protocol proves knowledge of an opening.

- $\mathsf{Init}(c, (x \in [-B, B], r \in [0, S]))$: $m \xleftarrow{\$} [0, BCL], s \xleftarrow{\$} [0, SCL]$; outputs $d = g^m h^s$.
- $\mathsf{Chall}()$: outputs $\gamma \xleftarrow{\$} [0, C]$
- $\mathsf{Resp}(\gamma)$: sets $z = m + \gamma \cdot x, t = s + \gamma \cdot r$. Outputs (z, t)
- $\mathsf{Verify}(d, \gamma, z, t)$: checks $|z| \leq BCL$ and $g^z h^t = d \cdot c^\gamma$.

The first verification check succeeds with overwhelming probability since the probability that the random m is too close to BCL is small. The second check succeeds due to

$$g^z h^t = g^{m+\gamma \cdot x} h^{s+\gamma \cdot r} = g^m h^s \cdot (g^x h^r)^\gamma = d \cdot c^\gamma.$$

Further, lemma 8 also implies that z, t hide the witnesses x, r statistically and using $d = g^z h^t \cdot c^{-\gamma}$, a valid transcript can be computed for a given challenge γ. Thus, the scheme honest-verifier is zero-knowledge. The following soundness argument shows how to extract correct openings.

First, let $(d, \gamma, z, t), (d, \gamma', z', t')$ be two accepting transcripts with $\gamma \neq \gamma'$. Without loss of generality, we assume that $\gamma' > \gamma$. We denote $\overline{z} = z' - z, \overline{t} = t' - t$ and $\overline{\gamma} = \gamma' - \gamma$. We know that $g^{z'-z} h^{t'-t} = c^{\gamma'-\gamma}$ which directly implies $g^{\overline{z}/\overline{\gamma}} h^{\overline{t}/\overline{\gamma}} = c$. Thus, $\gamma^* = \overline{\gamma}, r^* = \overline{t}/\overline{\gamma}, y^* = \overline{z}$ and $x^* = \lfloor \frac{y^*}{\gamma^*} \rceil$ is a valid opening for c. Note that the size checks are satisfied:

$$|\gamma^*| \leq C, \qquad |y^*| \leq 2BCL \leq U/C.$$

Note that we know that x^* is short because γ^* and y^* are short, so the above protocol can already be seen as range proof that guarantees that the committed value lies in $[-2BCL, 2BCL]$. Nonetheless, this is not very satisfying yet because the slackness of $2CL = 2^{2\lambda+1}$ is very large. But the shortness of the extracted values can be used to argue in $\mathbb{Z}$ instead of $\mathbb{Z}_q$ which opens the door for more sophisticated arguments.

On Retaining Homomorphism. If the original scheme is homomorphic, the transformation retains (restricted) homomorphic properties. Firstly, if the commitments are generated honestly, the homomorphic property is retained as long as the homomorphic calculation is performed inside the bound U/C of the scheme. In case of dishonest commitments, the scheme still retains a more limited form of homomorphic properties.

If the scheme com allows for addition of constants to the committed value, the homomorphic property is retained up to overflow over the bound U/C. To illustrate, let $\vec{t} \in \mathbb{Z}_q^n$ be some constant

and c a commitment to message $\vec{m} = \lfloor \vec{y}/\gamma \rceil$ with opening $(r, \gamma, \vec{y})$. Note that c commits to $\vec{y}/\gamma$ modulo q with respect to com and we can use the homomorphic operations. We have

$$(\vec{y}/\gamma) + \vec{t} = \vec{y}/\gamma + (\vec{t} \cdot \gamma)/\gamma = (\vec{y} + \vec{t} \cdot \gamma)/\gamma \mod q$$

and $\lfloor \frac{\vec{y}+\vec{t}\cdot\gamma}{\gamma} \rceil = \lfloor \vec{y}/\gamma \rceil + \vec{t} = \vec{m} + \vec{t}$. So the result of the homomorphic operation is actually exact because the additional operand does not introduce an additional error term. Note that for the opening to be correct, the norm $\left\| \vec{y} + \vec{t} \cdot \gamma \right\|_\infty$ needs to be smaller than U/C. So, enough space needs to be guaranteed to perform homomorphic operations. The analysis for retaining multiplicative homomorphic properties for small constants is similar.

In the case of additive and multiplicative homomorphisms between dishonest commitments, there are some small error terms and thus, the properties do not translate as directly. For homomorphic properties between dishonest commitments, the analysis is a bit more complicated and properties do not transfer directly. We illustrate this for the additive homomorphism. Let com be additively homomorphic, so there exists a function $\oplus$ for all commitments c_0, c_1 to messages $\vec{m}_0, \vec{m}_1 \in \mathsf{com}.\mathcal{M}_{\mathrm{com}}$ with openings r_0, r_1 respectively, it holds that:

$$c = \mathsf{com.Commit}(\vec{m}_0 + \vec{m}_1, r_0 \oplus r_1) = c_0 \oplus c_1.$$

Let c_0, c_1 be commitments with messages m_0, m_1 and valid openings $(r_0, \gamma, \vec{y}_0), (r_1, \gamma, \vec{y}_1)$ respectively with respect to the scheme $\overline{\mathsf{com}}$. We set $c = c_0 \oplus c_1$. Since the commitments c_0 and c_1 are valid commitments to messages $\vec{y}_0/\gamma \ (q)$ and $\vec{y}_1/\gamma \ (q)$ with respect to the scheme com, c is a commitment to $\vec{y} = (\vec{y}_0 + \vec{y}_1)/\gamma \ (q)$ with opening $r_0 \oplus r_1$. The bound and non-zero check for γ succeed by definition and if $\|\vec{y}_0 + \vec{y}_1\|_\infty < U/C$, then $\vec{y}$ passes the check as well.

The commitment will commit to $m = \lfloor \frac{\vec{y}_0+\vec{y}_1}{\gamma} \rceil$, where the division is performed in $\mathbb{R}^n$. We can write $\frac{\vec{y}_i}{\gamma} = \vec{m}_i + \vec{\mu}_i$ for $-0.5 \leq \vec{\mu}_i < 0.5$. With this, we have

$$\vec{m} = \vec{m}_0 + \vec{m}_1 + \lfloor \vec{\mu}_0 + \vec{\mu}_1 \rceil.$$

So the homomorphic calculation is correct, if $\vec{\mu}_0 + \vec{\mu}_1$ rounds to 0 for each component. This is not guaranteed for all possible openings, so the homomorphic property does not translate directly in the transformation without further care. Also, in case $\vec{y}_0 + \vec{y}_1$ is larger than U/C, the result of the homomorphic operation results in commitments that do not commit to the "desired" value. For example, let $y_0 + y_1 = 1/3 \mod q$ with (non-zero) representative of $1/3 > U/C$. The opening $(d_0 \oplus d_1, 3, 1)$ with message 0 will be a valid opening for the commitment. Thus, the use of homomorphic operations in $\overline{\mathsf{com}}$ need to be performed with attention to the specific setting.

For range proofs, the homomorphism with small constants can be used to prove the 3-square decomposition of the integer and the complications from multiplicative and the additive homomorphic error terms can be balanced out such that we can still prove the relation with the homomorphic property of the underlying schemes.

Ensuring Membership of an Interval. We use the 3 square decomposition in order to show membership of $[0, B]$. This can be extended to a range proof for interval $[a, b]$ by setting $B = b - a$. Since $\overline{\mathsf{com}}$ allows for addition of constants, the prover can show $x - a \in [0, B] \implies x \in [a, b]$. Note that the values still need to lie inside the given bounds.

We are using the 3 square decomposition to show that $x \in [0, B]$. Since the extracted x is a rounded fraction, we still need to ensure that the decomposition shows the desired range membership.

Lemma 13 (Three Square for Rounded Fractions). *Let $n, d \in \mathbb{Z}$ and $x = \lfloor \frac{n}{d} \rceil, \{x_i\}_{i=1..3} \in \mathbb{Q}$ and $B \geq 1$. Then:*

$$1 + 4\frac{n}{d}\left(B - \frac{n}{d}\right) = \sum_{i=1}^{3} x_i^2 \implies x \in [0, B].$$

Proof. A simple calculation shows that $\frac{n}{d} \in [\frac{1}{2}(B - \sqrt{B^2+1}), \frac{1}{2}(B + \sqrt{B^2+1})]$. This interval can further be bounded as follows:

$$\frac{1}{2}\left(B + \sqrt{B^2+1}\right) = \frac{1}{2}B\left(1 + \sqrt{1 + \tfrac{1}{B^2}}\right) \leq \frac{1}{2}B\left(1 + 1 + \frac{1}{2}\tfrac{1}{B^2}\right) = B + \tfrac{1}{4B}$$

where we use the Taylor expansion of $\sqrt{1+\varepsilon} = 1 + \frac{\varepsilon}{2} - O(\varepsilon^2) \leq 1 + \frac{\varepsilon}{2}$ for $0 \leq \varepsilon \leq 1$. A similar computation for the left bound shows that the 3-squares decomposition implies $\frac{n}{d} \in [-\frac{1}{4B}, B + \frac{1}{4B}]$. Since $B \geq 1$, we find $\frac{n}{d} \in [-\frac{1}{4}, B + \frac{1}{4}]$. Rounding leads to the desired result.

Further Properties. Our adapted commitment scheme and range proofs have additional useful properties.

Remark 2 (RP for com*).* For denominator $\gamma = 1$, $\overline{\mathsf{com}}$ coincides with com. Under this precondition, our range proofs establish $x \in [0, B]$ for also com-commitments.

Remark 3 (Positivity). Our proofs show $x \in [0, B]$. However, in many applications, proofs of positivity ($x \geq 0$) suffice. That is, B could be made into a zero-knowledge threshold (used for masking only), so that for $x > B$ no zero-knowledge guarantees hold.[18] This change is achieved by proving $1 + 4x = \sum_{i=1}^{3} x_i^2$. Now, soundness guarantees $x \in [0, \frac{q-1}{2}]$.

Remark 4 (Denominators). A closer look at the soundness proof shows, that if $x = \frac{m}{d}$ (without rounding) with $d \in [1, C]$ and m is small enough (e.g. $m \in [0, BL]$), then this leads to a rejection with probability $1 - \frac{1}{d}$. Thus, the larger d, the less likely will a (malicious) verifier succeed. This neither prevents "fractional" x nor do we know if m is small enough, so rounding is essential.

5 Range Proof in a DLOG Setting

5.1 Overview

In this section, we present the range proof in the setting of a group $\mathbb{G}$ with prime order q under the DLOG (or DLSE) assumption.[19] As basis, we use Pedersen commitments Ped, which we transform in a bounded rational commitment schemes $\overline{\mathsf{Ped}}$ as in section 4.1. Recall that the difference of Ped and $\overline{\mathsf{Ped}}$ is mostly in the interpretation of the committed values.

Our protocol reuses the structure of existing range proofs based on Pedersen commitments in the RSA setting (see [Lip03; Gro05; CPP17]). For a given commitment $c = g^x h^r$, the prover computes the square decomposition $1 + 4(b-x)x = \sum_{i=1..3} x_i^2$ and lets $x_0 = b - x$. Thus, we prove $1 + 4x_0 x = \sum_{i=1..3} x_i^2$. Note that all x_i are in the range $[0, B]$. The prover commits to $c_i = g^{x_i} h^{r_i}$ for some randomly sampled r_i for $i \in [1, 3]$, and sets $c_0 = g^b c^{-1}$. For a proof of knowledge of x_i, he computes mask commitments $d_i = g^{m_i} h^{s_i}$ (and an additional "garbage" term d), and sends them to the verifier. After receiving the challenge γ, the prover reveals $z_i = m_i + \gamma x_i$ and $t_i = s_i + \gamma r_i$ and the verifier can check whether the equation $g^{z_i} h^{t_i} = c_i^{\gamma} d_i$ holds (and an equation for the square decomposition).[20] The verifier checks the proof of knowledge and accepts only if z_i and t_i are small. As usual, if the prover can answer two different challenges $\gamma, \tilde{\gamma}$, openings can be extracted. These openings are $x_i = \frac{z_i - \tilde{z}_i}{\gamma - \tilde{\gamma}}$ with short nominator and denominator, and they satisfy the square decomposition (or DLOG is broken). This shows soundness (for $\overline{\mathsf{Ped}}$ openings), Furthermore, as we sketched in the introduction, when small exponents are used for the masking term h^y, and by adjusting the parameters, soundness can actually be proven *statistically*. In our parameter choice, however, we will optimize for efficiency and focus on computational soundness.

For zero-knowledge, the witness is blinded by the masks m_i. Since the m_i's must be small (hence are not uniform in $\mathbb{Z}_q$), we do not get perfect zero-knowledge. However, $x_i + m_i$ still statistically hides x_i. This is enough to establish (statistical) zero-knowledge by the usual "simulation by execution in reverse". The construction and proof is somewhat complicated by using *small* exponents for the masking term h^y, which consequently must be masked itself.

[18] In fact, masking and hence zero-knowledge degrades gracefully in the size of x.

[19] The optimization of the Pedersen commitment scheme with short exponents relies on the SEI, which for relevant ranges is equivalent to DLSE.

[20] In the scheme, we use a hash function to avoid having to send the mask commitments to the verifier to save space.

5.2 Parameters

Let $\mathsf{pp} = (g, h, q)$ be the public parameters of the commitment scheme Ped in group $\mathbb{G}$ with order q. Note that, for transparent setup, we should include the coins for sampling g and h, however, since we assume invertible sampling (e.g. simply interpreting a bitstring as a group element), we leave this requirement implicit in the following. Let $\mathsf{H} : \{0,1\}^* \mapsto \{0,1\}^{2\lambda}$ be a collision resistant hash function, and let $[0, B]$ be the range with $B \geq 2$. Let $[0, C]$ be the challenge set. Let S be the size of small exponents in the SEI assumption, and let L be the growth factor of masked intervals due to additive noise, that is, masking $[0, B]$ results in $[0, BL]$. We define $U = 32B^2C^2L^2$ and note that it serves as an upper bound for the integers appearing in the security proof. In particular, we require $U < \frac{q-1}{2}$. The prover shows that he knows x, r committed in $c = g^x h^r = \overline{\mathsf{Ped}}.\mathsf{Commit}(x; r)$ and that $x \in [0, B]$. (Other commitments are interpreted as Ped.)

5.3 Scheme

The scheme $\mathsf{RP}_{\mathsf{Log}}$ follows the structure of the line of work [Lip03; Gro05; CPP17]. We adapt the scheme to the DLOG setting and apply our encoding technique.

- $\mathsf{RP}_{\mathsf{Log}}.\mathsf{Init}(c = g^x h^r, x \in [0, B], r \in [0, S])$:
  1. compute x_i s.t. $4x(B - x) + 1 = \sum_{i=1}^{3} x_i^2$
  2. Set $r_0 = -r, x_0 = B - x$
  3. Set $c_0 = c^{-1} g^B$
  4. Set $\forall i \in [1, 3] : r_i \xleftarrow{\$} [0, S], c_i = g^{x_i} h^{r_i}$
  5. Set $\forall i \in [0, 3] : m_i \xleftarrow{\$} [0, BCL], s_i \xleftarrow{\$} [0, SCL], d_i = g^{m_i} h^{s_i}$
  6. Set $\sigma \xleftarrow{\$} [0, 4SBCL], d = h^\sigma c^{4m_0} \prod_{i=1..3} c_i^{-m_i}$
  7. Set $\Delta = H(\{d_i\}_{i=0..3}, d)$
  8. Outputs $\{c_i\}_{i=1..3}, \Delta$
- $\mathsf{RP}_{\mathsf{Log}}.\mathsf{Chall}()$: outputs $\gamma \xleftarrow{\$} [0, C]$
- $\mathsf{RP}_{\mathsf{Log}}.\mathsf{Resp}(\gamma)$:
  1. Sets $\forall i \in [0, 3] : z_i = m_i + \gamma \cdot x_i, t_i = s_i + \gamma \cdot r_i$
  2. Sets $\tau = \sigma + \gamma(\sum_{i=1..3} x_i r_i + 4x_0 r_0)$
  3. Outputs $\{z_i, t_i\}_{i=0..3}, \tau$
- $\mathsf{RP}_{\mathsf{Log}}.\mathsf{Verify}(\{c_i\}_{i=1..3}, \Delta, \gamma, \{z_i, t_i\}_{i=0..3}, \tau)$:
  1. Compute $c_0 = c^{-1} g^B$
  2. Compute $\forall i \in [0, 3] : f_i = g^{z_i} h^{t_i} c_i^{-\gamma}$
  3. Compute $f = h^\tau \cdot g^\gamma \cdot c^{4z_0} \cdot \prod_{i=1..3} c_i^{-z_i}$
  4. Check $\Delta = H(\{f_i\}_{i=0..3}, f)$
  5. Check $z_i \in [0, BC(L + 1)]$

Note that any interval $[0, T]$, where term T contains S, may be replaced by $[0, \max(q - 1, T)]$, as these masks only serve zero-knowledge and do not affect soundness, hence wraparound is not a problem. In particular, the scheme is correct, sound and HVZK if $S = q - 1$.

Theorem 1. *The scheme* $\mathsf{RP}_{\mathsf{Log}}$ *is perfectly correct, i.e. has correctness error* 0.

Proof. The three-square decomposition $\{x_i\}_{i=1..3}$ can be efficiently computed by lemma 1. We have $x, x_0, \ldots x_3 \in [0, B]$, $\gamma \in [0, C]$, $m_i \in [0, BCL]$, and hence $z_i = \gamma x_i + m_i \in [0, BC(L + 1)]$. Thus, the size check for z_i never fails. Also for $i \in [0, 3]$:

$$\begin{aligned} F_i &= g^{z_i} h^{t_i} c_i^{-\gamma} \\ &= g^{m_i + \gamma \cdot x_i} h^{s_i + \gamma \cdot r_i} (g^{x_i} h^{r_i})^{-\gamma} \\ &= g^{m_i} h^{s_i} (g^{x_i} h^{r_i})^{\gamma} (g^{x_i} h^{r_i})^{-\gamma} = d_i \end{aligned}$$

and further:

$$
\begin{aligned}
f &= h^{\tau} \cdot g^{\gamma} \cdot c^{4z_0} \cdot \prod_{i=1..3} c_i^{-z_i} \\
&= h^{\tau} \cdot g^{\gamma} \cdot c^{4z_0} \cdot g^{-\gamma \sum_{i=1..3} x_i^2} \cdot h^{-\gamma \sum_{i=1..3} r_i \cdot x_i} \cdot \prod_{i=1..3} g^{-x_i m_i} \cdot h^{-r_i \cdot m_i} \\
&= h^{\tau} \cdot g^{\gamma} \cdot c4z_0 \cdot g^{-\gamma \cdot (4x(B-x)+1)} \cdot h^{-\gamma \sum_{i=1..3} r_i \cdot x_i} \cdot \prod_{i=1..3} c_i^{-m_i} \\
&= h^{\tau} \cdot g^{\gamma} \cdot (g^{4x} h^{4r})^{m_0 + \gamma \cdot x_0)} \cdot g^{-\gamma \cdot (4x(B-x)+1)} \cdot h^{-\gamma \sum_{i=1..3} r_i \cdot x_i} \cdot \prod_{i=1..3} c_i^{-m_i} \\
&= h^{\tau} \cdot c^{4m_0} \cdot g^{\gamma \cdot (4x(B-x)+1)} \cdot h^{\gamma \cdot 4rx_0} \cdot g^{-\gamma \cdot (4x(B-x)+1)} \cdot h^{-\gamma \sum_{i=1..3} r_i \cdot x_i} \cdot \prod_{i=1..3} c_i^{-m_i} \\
&= h^{\tau} \cdot h^{\gamma \cdot (4r_0 x_0 - \sum_{i=1..3} r_i \cdot x_i)} \cdot c^{4m_0} \cdot \prod_{i=1..3} c_i^{-m_i} \\
&= h^{\sigma} \cdot c^{4m_0} \cdot \prod_{i=1..3} c_i^{-m_i} = d
\end{aligned}
$$

Theorem 2. *The proof system* $\mathsf{RP}_{\mathsf{Log}}$ *is HVZK with simulation error* $9/L$*. If* $S = q - 1$*, this holds against unbounded adversaries.*

More precisely, for every HVZK adversary $\mathscr{A}$*, there is a* SEI *adversary* $\mathscr{B}$ *with roughly the same running time as* $\mathscr{A}$*, such that* $\mathsf{Adv}^{\mathrm{hvzk}}_{\mathscr{A}} \leq 9/L + 4\mathsf{Adv}^{\mathrm{sei}}_{\mathscr{B}}$*.*

Proof. First, recall that $S = q - 1$, then $\mathsf{Adv}^{\mathrm{sei}}_{\mathscr{B}} = 0$. Hence SEI is unconditionally secure and the simulation error holds against unbounded adversaries. With this, we turn to the general security reduction.

We modify the distribution of valid transcripts in four indistinguishable steps such that the last game does not require a witness as input. The simulator $\mathsf{Sim}_{\mathsf{ZK}}$ defined by game 4 fulfills the requirements of the zero-knowledge definition.

Game 1:
Outputs an unmodified transcript from an interaction of an honest verifier and prover from the definition:

- Compute x_i s.t. $4x(B-x)+1 = \sum_{i=1}^{3} x_i^2$
- Set $r_0 = -r, x_0 = B - x$
- Set $c_0 = c^{-1} g^B$
- Set $\forall i \in [1,3] : r_i \xleftarrow{\$} [0, S], c_i = g^{x_i} h^{r_i}$
- Set $\forall i \in [0,3] : m_i \xleftarrow{\$} [0, BCL], s_i \xleftarrow{\$} [0, SCL], d_i = g^{m_i} h^{s_i}$
- Set $\sigma \xleftarrow{\$} [0, 4SBCL], d = h^{\sigma} c^{4m_0} \prod_{i=1..3} d_i^{-m_i}$
- Set $\Delta = H(\{d_i\}_{i=0..3}, d)$
- $\forall i \in [0,3] : z_i = m_i + \gamma \cdot x_i, t_i = s_i + \gamma \cdot r_i$
- $\tau = \sigma + \gamma(\sum_{i=1..3} x_i r_i - 4x_0 r_0)$
- Outputs $\{\{c_i\}_{i=1..3}, \Delta, \{z_i, t_i\}_{i=0..3}, \tau\}$

for given $\gamma \in [0, C], c = g^x h^r, x \in [0, B], r \in [0, S]$.

Game 2:
Rewrites the mask commitments $\{d_i\}_{i=1..3}, d$ as in the verification check.:

- Compute x_i s.t. $4x(B-x)+1 = \sum_{i=1}^{3} x_i^2$
- Set $r_0 = -r, x_0 = B - x$
- Set $c_0 = c^{-1} g^B$
- Set $\forall i \in [1,3] : r_i \xleftarrow{\$} [0,S], c_i = g^{x_i} h^{r_i}$
- Set $\forall i \in [0,3] : m_i \xleftarrow{\$} [0, BCL], s_i \xleftarrow{\$} [0, SCL]$
- Set $\sigma \xleftarrow{\$} [0, 4SBCL]$
- $\forall i \in [0,3] : z_i = m_i + \gamma \cdot x_i, t_i = s_i + \gamma \cdot r_i$
- $\tau = \sigma + \gamma(\sum_{i=1..3} x_i r_i - 4x_0 r_0)$
- Set $\forall i \in [0,3] : \boxed{d_i = g^{z_i} h^{t_i} c_i^{-\gamma}}$
- Set $\boxed{d = h^{\tau} \cdot g^{\gamma} c^{4z_0} \cdot \prod_{i=1..3} c_i^{-z_i}}$
- Set $\Delta = H(\{d_i\}_{i=0..3}, d)$
- Outputs $\{\{c_i\}_{i=1..3}, \Delta, \{z_i, t_i\}_{i=0..3}, \tau\}$

for given $\gamma \in [0, C], c = g^x h^r, x \in [0, B], r \in [0, S]$.

Claim. Game 1 and Game 2 are identically distributed.

Proof. As shown in the correctness proof, it holds that $g^{m_i} h^{s_i} = g^{z_i} h^{t_i} c_i^{-\gamma}$ for $i \in [0,3]$ and $h^{\sigma} d_a^{m_0} \prod_{i=1..3} d_i^{-m_i} = h^{\tau} \cdot g^{\gamma} c^{4z_0} \cdot \prod_{i=1..3} c_i^{-z_i}$.

Game 3:
Removes the dependency on x in the zero-knowledge witnesses $t_i, z_i.\tau$ for $i \in [0,3]$:

- compute x_i s.t. $4x(B-x)+1 = \sum_{i=1}^{3} x_i^2$
- Set $r_0 = -r, x_0 = B - x$
- Set $c_0 = c^{-1} g^B$
- Set $\forall i \in [1,3] : r_i \xleftarrow{\$} [0,S], c_i = g^{x_i} h^{r_i}$
- $\forall i \in [0,3] : \boxed{z_i \xleftarrow{\$} [0, BCL], t_i \xleftarrow{\$} [0, SCL]}$
- $\boxed{\tau \xleftarrow{\$} [0, 4SBCL]}$
- Set $\forall i \in [0,3] : d_i = g^{z_i} h^{t_i} c_i^{-\gamma}$
- Set $d = h^{\tau} \cdot g^{\gamma} c^{4z_0} \cdot \prod_{i=1..3} c_i^{-z_i}$
- Set $\Delta = H(\{d_i\}_{i=0..3}, d)$
- Outputs $\{\{c_i\}_{i=1..3}, \Delta, \{z_i, t_i\}_{i=0..3}, \tau\}$

for given $\gamma \in [0, C], c = g^x h^r, x \in [0, B], r \in [0, S]$.

Claim. Game 2 and Game 3 have statistical distance at most $9/L$.

Proof. The claim follows from the bounds

$$\begin{aligned} 0 \leq & x_i \leq B, \\ 0 \leq & r_i \leq S, \\ -4 \cdot BS^2 \leq -4x_0 r_0 \leq & \sum_{i=1..3} x_i r_i - 4x_0 r_0 \leq \sum_{i=1..3} x_i r_i \leq 4 \cdot BS \end{aligned}$$

and our choice of masking. Namely, lemma 8 asserts a statistical distance of L^{-1} per application of masking.

Game 4:
Removes the dependency on x from the commitments:

- Set $c_0 = c^{-1}g^B$
- $\forall i \in [0,3] : z_i \xleftarrow{\$} [0, BCL], t_i \xleftarrow{\$} [0, SCL]$
- $\tau \xleftarrow{\$} [0, 4SBCL]$
- Set $\forall i \in [1,3] : r_i \xleftarrow{\$} [0,S], \boxed{c_i \xleftarrow{\$} g^0 h^{r_i}}$
- Set $\forall i \in [0,3] : d_i = g^{z_i} h^{t_i} c_i^{-\gamma}$
- Set $d = h^\tau \cdot g^\gamma c^{4z_0} \cdot \prod_{i=1..3} c_i^{-z_i}$
- Set $\Delta = H(\{d_i\}_{i=0..3}, d)$
- Outputs $\{\{c_i\}_{i=1..3}, \Delta, \{z_i, t_i\}_{i=0..3}, \tau\}$

for given $\gamma \in [0, C], c = g^x h^r$.

Claim. Game 3 and Game 4 are indistinguishable under the hiding property of the commitment scheme Ped.

Proof. We provide an algorithm $\mathscr{B}$ breaking the hiding property of Ped using a distinguisher $\mathscr{A}$ of Game 3 and Game 4. First, $\mathscr{B}$ sets up the parameters of the range proof with the $\mathsf{pp} = (g, h)$ received from the hiding challenger $\mathscr{C}$. Then it computes x_i s.t. $4x(B-x)+1 = \sum_{i=1}^3 x_i^2$ and sends $m_0 = (x_1, x_2, x_3)$ and $m_1 = (0,0,0)$ to $\mathscr{C}$. It then constructs the transcript of a proof as before utilizing the received commitments $\{c_i\}_{i=1..3}$ and sends the transcript to $\mathscr{A}$. After receiving b from $\mathscr{A}$ it forwards the bit to the challenger $\mathscr{C}$.

Now, if challenger committed to m_0, the output of $\mathscr{B}$ is distributed as in game 3 and otherwise as in game 4. Thus if $\mathscr{A}$ has non-negligible advantage, $\mathscr{B}$ breaks the hiding property of the commitment scheme. Note that we argue utilize the hiding property on vectors of messages which is implied by the hiding property of definition 3 for a constant vector size.

Theorem 3. *Suppose $L \geq 32$. The range proof $\mathsf{RP}_{\mathsf{Log}}$ for $[0, B]$ is 2-special sound with knowledge error $\frac{1}{(C+1)}$ under DLOG and CRHF assumptions.*
More precisely, for every adversary $\mathscr{A}$ with strict running time T there are adversaries $\mathscr{B}_1, \mathscr{B}_2$ with expected running time roughly $2T$ and $\mathsf{Adv}^{\mathrm{ke}}_{\mathscr{A}} \leq \frac{1}{(C+1)} + \mathsf{Adv}^{\mathrm{dlog}}_{\mathscr{B}_1} + \mathsf{Adv}^{\mathrm{crhf}}_{\mathscr{B}_2}$.

Proof. Assume we have two accepting transcripts for distinct challenges $\gamma \neq \tilde{\gamma}$ with witnesses z_i, t_i, τ and $\tilde{z_i}, \tilde{t_i}, \tilde{\tau}$ respectively. Without loss of generality, say $\gamma > \tilde{\gamma}$. We show that either we obtain a valid witness, or we break DLOG or collision resistance.

By collision resistance of H, we have $d = f = \tilde{f}$ and $\forall i \in [0,3] : d_i = f_i = \tilde{f}_i$. Denote by $\overline{a}$ the difference of $a - \tilde{a}$ for $a \in \{z_i, t_i, \tau\}$. From $f_i = \tilde{f}_i$ we find

$$g^{z_i} h^{t_i} c_i^{-\gamma} = g^{\tilde{z_i}} h^{\tilde{t_i}} c_i^{-\tilde{\gamma}} \iff g^{\overline{z_i}} h^{\overline{t_i}} = c_i^{\overline{\gamma}} \iff g^{\overline{z_i}/\overline{\gamma}} h^{\overline{t_i}/\overline{\gamma}} = c_i.$$

Thus for all $i \in [1,3]$, we have valid openings $x_i = \overline{z_i}/\overline{\gamma}$ and $r_i = \overline{t_i}/\overline{\gamma}$ for commitment c_i. For c_0, we obtain $c = g^{(\overline{\gamma} \cdot B - \overline{z_0})/\overline{\gamma}} h^{-\overline{t_0}/\overline{\gamma}}$ and therefore $x_0 = \overline{z_0}/\overline{\gamma}$ and $r_0 = \overline{t_0}/\overline{\gamma}$ is an opening to $c^{-1}g^B$. Moreover $x = B - \overline{z_0}/\overline{\gamma} = B - x_0$ is the committed value in c.

Now we turn to the square decomposition. We have

$$\begin{aligned}
f = \tilde{f} &\implies h^{\overline{\tau}} \cdot g^{\overline{\gamma}} \cdot c^{4\overline{z_0}} = \prod_{i=1..3} c_i^{\overline{z_i}} \\
&\implies h^{\overline{\tau}} \cdot g^{\overline{\gamma}} \cdot g^{4(B - \overline{z_0}/\overline{\gamma})\overline{z_0}} \cdot h^{4r \cdot \overline{z_0}} = \prod_{i=1..3} g^{x_i \cdot \overline{z_i}} h^{r_i \cdot \overline{z_i}} \\
&\implies g^{\overline{\gamma}} \cdot g^{4(B - \overline{z_0}/\overline{\gamma})\overline{z_0}} \cdot \prod_{i=1..3} g^{-x_i \cdot \overline{z_i}} = h^{-4r \cdot \overline{z_0}} \cdot h^{-\overline{\tau}} \cdot \prod_{i=1..3} h^{r_i \cdot \overline{z_i}} \\
&\implies g^{\overline{\gamma} + 4(B - \overline{z_0}/\overline{\gamma})\overline{z_0} - \sum_{i=1..3} x_i \cdot \overline{z_i}} = h^{-4r \cdot \overline{z_0} - \overline{\tau} + \sum_{i=1..3} r_i \cdot \overline{z_i}}.
\end{aligned}$$

Under the DLOG assumption (or statistically, when the exponent of h remains small enough), this forces

$$\begin{aligned}
&\overline{\gamma} + 4(B - \overline{z_0}/\overline{\gamma})\overline{z_0} - \sum_{i=1..3} x_i \cdot \overline{z_i} = 0 \mod q \\
\implies &\overline{\gamma} + 4(B - \overline{z_0}/\overline{\gamma})\overline{z_0} = \sum_{i=1..3} \overline{z_i}^2/\overline{\gamma} \mod q \\
\implies &\overline{\gamma}^2 + 4(\overline{\gamma} \cdot B - \overline{z_0})\overline{z_0} = \sum_{i=1..3} \overline{z_i}^2 \mod q
\end{aligned}$$

The final equality holds over the integers, because all values are small enough so that there is no wrap-around. More precisely: Let $K = BC(L+1)$ be the maximal (accepting) value of $|z_i|$. For the right hand side, $|\overline{z_i}| \leq |z_i| + |\tilde{z}_i| \leq 2K$ and hence $\sum_{i=1..3} \overline{z_i}^2 \leq 16K^2 \leq U < \frac{q-1}{2}$. Rewrite the left hand side as $\overline{\gamma}^2 + 4\overline{\gamma}B\overline{z_0} - \overline{z_0}^2$. Shortness follows from $|\gamma|B \leq K$ and thus $K^2 + 8K^2 + 16K^2 \leq 25K^2 \leq U < \frac{q-1}{2}$. Here we use that $25K^2 = 25(BC(L+1))^2 \leq 32(BCL)^2 = U$ since $L \geq 32$.

Since the equality holds over the integers, after dividing by $\overline{\gamma}^2$ it holds over $\mathbb{Q}$. Using $\overline{z_0} = \overline{\gamma}(B-x)$, we see that $\overline{\gamma}^2 + 4\overline{\gamma}x(\overline{\gamma}B - \overline{\gamma}x) = \sum_{i=1}^3 \overline{\gamma}^2 x_i^2$ and hence $1 + 4x(B-x) = \sum_{i=1}^3 x_i^2$ for $x = B - \frac{\overline{z_0}}{\overline{\gamma}}$. Now, lemma 13 finishes the proof. (Note that we extracted a valid opening for c.)

5.4 Optimizations

We discuss some optimizations to either reduce the proof size or the group size.

Rejection sampling for smaller group size. In $\mathsf{RP}_{\mathsf{Log}}$, we hide the values $\gamma \cdot x_i \in [0, BC]$ by an additive uniformly random mask $z \in [0, BCL]$. So the masking has an overhead of $\log(L)$ bits. By using rejection sampling for masking, as used in the lattice setting, this overhead can be traded for a (small) correctness error. For this, we apply lemma 11 instead of lemma 8. That is, we choose the mask from a discrete Gaussian distribution with large enough standard deviation σ_x, and the prover aborts in Resp with (small) probability.

More concretely: Let the parameters for rejection sampling be standard deviation $\sigma_x = \alpha \cdot BC$ and $M = e^{13.3/\alpha + 1/(2\alpha^2)}$ for some α. Let $k = \sqrt{2\lambda}$ and let $L' = \lceil k\alpha \rceil$. Then the probability that the mask $m \leftarrow D_{\sigma_x}$ is too large (and causes verification to abort) is $O(2e^{-k^2/2}) = \mathrm{negl}(\lambda)$ by lemma 10. The protocol is adapted as follows[21]:

- In Init, sample $m_i \leftarrow D_{\sigma_x}$ for $i \in [0,3]$ (instead of $m_i \leftarrow [0, BCL']$).
- In Resp, abort with probability $1 - \min\left(\frac{D_{\sigma_x(z_i)}}{M \cdot D_{\gamma \cdot x_i, \sigma_x(z_i)}}, 1\right)$ for $i \in [0,3]$,
- In Verify, check $|z_i| \leq BC(L'+1)$ for $i \in [0,3]$ instead of $z_i \in [0, BC(L'+1)]$.

Since $|m_i| \leq BCL'$ (and thus $|z_i| \leq BC(L'+1)$) with overwhelming probability, the completeness is mostly affected by aborting in Resp. For the concrete value $\alpha = 256$ which implies $M \approx 1.05$, the abort probability is very small (roughly 0.05). The statistical distance between honest masking and "simulated" masked values is at most $\delta = 2^{-120}$, by lemma 11. Using this property the HVZK simulator is easily adapted and achieves simulation error $4\delta + 5L^{-1}$. (Note that s_i and σ are sampled as before.) The soundness proof uses L' but is otherwise unchanged.

To achieve non-negligible completeness, the protocol needs to be repeated, increasing computation and communication. For the Fiat–Shamir transformation, only computation increases.

Lastly, note that $2U = 32(BCL')^2$ is a lower bound on the group size q. With rejection sampling, we can choose smaller L', and hence smaller q. One can use rejection sampling for the masks σ and s_i as well, but these do not affect the group size, only the communication (and the simulation error). More concretely, let $\sigma_r = \alpha \cdot SCL$ and further modify the protocol as follows:

- In Init choose $s_i \leftarrow D_{\sigma_r}$ for $i \in [0,3]$.
- In Resp abort with probability $1 - \min\left(\frac{D_{\sigma_r(t_i)}}{M \cdot D_{\gamma \cdot r_i, \sigma_r} t_i)}, 1\right)$ for $i \in [0,3]$.

This results in a size of $|t_i| \leq SCL'$. Also applying this to σ yields $|\tau| \leq 4SBCL'$.

[21] For more details on the technique and the proof of security, we refer to the range proof in the lattice setting of section 6. It uses rejection sampling for masking the randomness of the commitment scheme.

Soundness amplification for smaller group size. The soundness error of the scheme is $1/(C+1)$, and since C affects U and hence the group size, decreasing it allows smaller groups. However, to achieve negligible soundness error, multiple iterations are required, namely $\lambda/\log(C)$ iterations for a soundness error of $2^{-\lambda}$. Note that the commitments c_i only need to be sent in the first repetition and can be reused in the following ones.

5.5 Efficiency

Now, we discuss the concrete efficiency of the (optimized) construction. For this we instantiate the parameters as follows: $B \in \{2^{32}, 2^{64}\}, C = 2^{\lambda}, L = 2^{\lambda}, S = 2^{2\lambda}, L' = \lceil 256\sqrt{2\lambda} \rceil$. Note that the group size is $64(BCL)^2$ or, using rejection sampling for masking x_i, $64(BCL')^2$.

Optimized Non-Interactive Range Proof with Standard Soundness. An entire proof of the scheme consists of elements $\{c_i\}_{i=1..3}, \Delta, \{z_i, t_i\}_{i=0..3}, \tau$. The proof sizes for ranges 32 and 64 bit are summarized in Table more detailed comparison with Bulletproof is given in Table 1 in the introduction, for $\lambda \in \{80, 128\}$. They were computed with a Python script presented in appendix C. The following remarks of section section 5.4 were applied on the scheme:

- Rejection sampling for m_i, s_i, σ,
- Fiat-Shamir transformation for non-interactivity.

Since $z_i \in [0, BCL']$ with overwhelming probability, we calculate with this bound as opposed to the one from the scheme. Note that we use a challenge size of λ bits but for all examples, a standard group size can be achieved by utilizing a smaller challenge and repeating the proof the corresponding number of times. For $U = 32(BCL')^2$ the proof size in bit is

$$3(\log(2U)) + \lceil \lambda/\log(C) \rceil (2\lambda + 4(\log(BCL') + \log(SCL')) + \log(4SBCL'))$$

Concrete parameteres and proof sizes are given in the introduction in table 1.

Computational Efficiency. We count the computational efficiency of the optimized non-interactive range proof in multiplications of group elements. Multiplications in elliptic curves are generally expensive, whereas discrete Gaussian sampling and the computation of the 3 square decomposition are comparatively insignificant, since they have a $\mathcal{O}(1)$ and expected $\mathcal{O}(B^2/\log(B))$ complexity respectively [MW17; PS19]. The decomposition algorithm takes on average 30 ms for 500 bit values [PS19], thus accounting for the decrease in input size the decomposition will take less than 1 ms on average for 32 or 64 bit ranges [22]. Additionally, the proof might have to be aborted and rerun in case the constant M of lemma 11 is not chosen overwhelmingly close to 1. This can be ensured in exchange for slightly larger witnesses (and thus a slight increase in group size). The total probability of a rerun is about 65% with our chosen parameters of the optimized range proof. Thus, the expected number of group operations is about 1.54 of group operations of a single successful run. Note that if we only apply rejection sampling on the masks $\{m_i\}_{i=0..3}$, the protocol needs to be repeated with a probability of 18% while keeping the same group size.

An exponentiation with a k-bit value costs on average $1.5 \cdot k$ multiplications and we count inverting separately. Further, in order to speedup the calculation of a Pedersen commitment $c = g^x h^r$, we utilise the following trick. Let us assume that $x > r$ without loss of generality. We set $c = (gh)^r \cdot g^{x-r}$. Thus, the commitment is computed in $1.5 \cdot x + 1$ multiplications. Lastly, we also assume that $B < \lambda$. We disregard other possible optimizations for verification.

- The prover has to perform $2.31 \cdot (4 \log B + 8 \log S + 6 \log C + 7 \log L') + 30$ multiplications and 1.54 inversions.
- The verifier has to perform $4.5 \log B + 7 \log S + 13 \log C + 9 \log L' + 10$ multiplications and 6 inversions.

[22] The average runtime was calculated as follows: $30ms \cdot (64^2/log_2(64))/(500^2/log_2(500))$.

6 Range Proof in a Lattice Setting

6.1 Overview

We now establish a range proof protocol in the lattice setting. The setup and scheme mainly rely upon the techniques from [Yan+19]. In the lattice setting, we naturally deal with vectors $\vec{x} \in \mathbb{Z}_q^n$. Because our technique requires a large modulus q, performing one range proof at a time would waste a significant amount of space. Thus, we look at a more general batch setting, where a commitment to $\vec{x} \in \mathbb{Z}^n$ is given and we desire to prove $x_i \in [0, B_i]$ for $\vec{B} \in \mathbb{Z}^n$. We write shortly $\vec{x} \in [0, \vec{B}]$ for this statement.

We use the commitment scheme Lat defined in definition 13 and its corresponding bounded integer commitment scheme $\overline{\mathsf{Lat}}$ as basis for the proof. The proof of knowledge of an opening of a Lat commitment $\vec{c} = \boldsymbol{A} \cdot \vec{r} + (\vec{0} \parallel \vec{x})$ for $\vec{x} \in [0, \vec{B}], \vec{r} \xleftarrow{\$} D_\sigma^l$ is performed by masking the witnesses via $\vec{z} = \vec{m} + \gamma \cdot \vec{x}$ and $\vec{t} = \vec{s} + \gamma \cdot \vec{r}$ and sending a commitment $\vec{d} = \boldsymbol{A} \cdot \vec{s} + (\vec{0} \parallel \vec{m})$ of the masks in order to allow to verify the relation. Here, γ is a short challenge, $\vec{m} \in \mathbb{Z}_q^n$ hides $\gamma \cdot \vec{x}$ and the mask $\vec{s}$ is sampled from a discrete Gaussian distribution such that $\vec{t}$ is short enough to suffice as opening for the commitment and $\gamma \cdot \vec{r}$ is hidden according to lemma 11.

The verifier checks the relation via $\boldsymbol{A} \cdot \vec{t} + (\vec{0} \parallel \vec{z}) = \vec{d} + \gamma \cdot \vec{c}$ and the binding property is retained if the verifier checks that $\vec{t}$ is indeed short. In the soundness proof, the extractor can extract the opening of the commitments via $\vec{x} = (\vec{z'} - \vec{z}) \cdot (\gamma' - \gamma)^{-1}$ and $\vec{r} = (\vec{t'} - \vec{t}) \cdot (\gamma' - \gamma)^{-1}$ given transcripts τ, τ' of the proof with distinct challenges γ, γ'.

For our integer commitment technique, we further require the vector $\vec{z}$ to be short. Since $\vec{x}$ will be short component-wise, we can simply verify that $\vec{z}$ is short if $\vec{m}$ is chosen short but large enough to hide $\gamma \cdot \vec{x}$ according to lemma 8. In summary, with the above techniques we can show that we know openings of the commitment $\vec{c} \leftarrow \overline{\mathsf{Lat}}.\mathsf{Commit}(\vec{x}), \vec{c}_i \leftarrow \mathsf{Lat}.\mathsf{Commit}(\vec{x}_i)$. We still need to prove the relation

$$4\vec{x}(\vec{B} - \vec{x}) + \vec{1} = \sum_{i=1}^{3} \vec{x}_i^2 \tag{2}$$

for $\vec{x} \in [0, \vec{B}]$ and appropriate vectors $\vec{x}_i$. We utilize a commitment to $\vec{x}_0 = \vec{B} - \vec{x}$ which can be computed homomorphically. Now, the equation eq. (2) can be rewritten as follows:

$$4\vec{x}\vec{x}_0 + \vec{1} - \sum_{i=1}^{3} \vec{x}_i^2 = 0 \tag{3}$$

Since the commitments are not sufficiently homomorphic to check the relation eq. (3) directly, we need to show the relation in a different manner. In our proof, we utilize a standard technique in lattice-based zero-knowledge proofs that allows to check the verification using the masked witnesses $\vec{z}_i = \vec{m}_i + \gamma \cdot \vec{x}_i$. The core observation is that since $\vec{z}_i$ contains $\vec{x}_i$, we can try to check the equation by replacing $\vec{x}_i$ with $\vec{z}_i$ in eq. (3). A simple calculation shows that if we also replace $\vec{1}$ with $\vec{\gamma}^2$ and $\vec{B}$ with $\gamma\vec{B}b$, and further interpret the vector

$$\vec{f} = 4 \cdot \vec{z}\vec{z}_0 + \vec{\gamma}^2 - \sum_{i=1..3} \vec{z}_i^2$$

as polynomial $\vec{f} = \vec{f}_2\gamma^2 + \vec{f}_1\gamma + \vec{f}_0 \in \mathbb{Z}_q[\gamma]$, the leading coefficient $\vec{f}_2$ is equal to the left side of eq. (3). Since $\vec{f}$ can be computed by the verifier, it suffices that the prover proves that $\vec{f}$ is a polynomial of degree 1. This can be done by committing to the garbage terms $\vec{f}_0$ and $\vec{f}_1$ in $\vec{c}_f = \mathsf{Lat}.\mathsf{Commit}(\vec{f}_1; \vec{r}_f)$ and $\vec{d}_f = \mathsf{Lat}.\mathsf{Commit}(\vec{f}_0; \vec{s}_f)$ respectively. Now if the prover sends $(\vec{c}_f, \vec{d}_f)$ and $\vec{t}_f = \vec{s}_f + \gamma \cdot \vec{r}_f$ to the verifier, he can verify whether indeed $\mathsf{Lat}.\mathsf{Commit}(\vec{f}; \vec{t}_f) = \vec{d}_f + \gamma\vec{c}_f$ using the additive homomorphic properties of Lat. This check ensures that $\vec{f}$ is a polynomial of degree 1. Revealing $\vec{t}$ does not reveal either opening if $\vec{s}$ is sampled by a discrete gaussian distribution according to lemma 11. Since the sampled $\vec{s}$ is short enough to suffice as opening for the commitment, the binding property is guaranteed. The commitments $\vec{c}_f, \vec{d}_f$ are sent in the first step and the polynomial $\vec{f}$ only has 3 roots. Since γ has to be one of them and it is chosen at random by the verifier in the next step, the prover can not cheat. Note that in the proof, we denote $\vec{f}$ as $\vec{z}_4$ in order to simplify the notation.

This concludes the proof of eq. (2) and we can extract the witnesses of the commitments via $\vec{x}_i = (\vec{z}'_i - \vec{z}_i) \cdot (\gamma' - \gamma)^{-1}$, where $\gamma' - \gamma$ and $\vec{z}'_i - \vec{z}_i$ are short. Since the extracted elements are short, the decomposition holds over the integers and subsequently, lemma 13 yields $\vec{x} \in [0, \vec{B}]$ after rounding.

Note that we chose this lattice setting due to complications when applying this technique to ring lattices (see appendix A for further details).

6.2 Parameters

Let $n \in \mathbb{N}$ and $\vec{B} \in \mathbb{Z}^n$ with $\vec{B} \geq \vec{2}$. Let $l_1, l_2, C \in \mathsf{poly}(\lambda)$ and $l := l_1 + n + l_2$. Further, let $\sigma \geq \sqrt{2l_2/\pi}$ and $\sigma_{\mathsf{rej}} \geq 2C \cdot \sqrt{l} \cdot \log(l) \cdot \sigma$ be positive integers. Let $M = e^{\frac{13.3}{\log(l)} + \frac{1}{2\log^2(l)}}$ and $T = 4\sqrt{l} \cdot (\sigma_{\mathsf{rej}} + C \cdot \sigma)$ and let $q = q_0^e$ for some prime q_0 and $e \in \mathbb{N}_+$. Let $\mathsf{pp} = \boldsymbol{A} \leftarrow \mathsf{Lat.Setup}(1^\lambda)$. Note that pp can easily be sampled transparently, as the two uniformly random matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ in $\boldsymbol{A}$ can be interpreted as the public coins of the sampling algorithm. Further, let $U = 32B^2C^2L^2 < \frac{q-1}{2}$ for $B = \left\|\vec{B}\right\|_\infty$ and $L \in \mathbb{N}$.

Hereby, the values $q, n, l_1, l_2, C, \sigma, T$ are the parameters of the commitment scheme Lat and $\overline{\mathsf{Lat}}$. In addition, the value U is the size requirement parameter of the scheme $\overline{\mathsf{Lat}}$. The value σ_{rej} is the standard deviation of the Gaussian distribution $D_{\sigma_{\mathsf{rej}}}$ used to mask the opening via rejection sampling with constant M (see lemma 11). The vector $\vec{B}$ defines the range $[0, B_i]$ for each performed range proof. The values B, L are bounds on the range and masking overhead respectively.

6.3 Scheme

The setup and scheme uses techniques from [Yan+19] but is adapted to proving the statement $\vec{1} + 4\vec{x}(\vec{B} - \vec{x}) = \sum_{i=1}^{3} \vec{x}_i^2$ for $\vec{x} \in [0, \vec{B}]$ and appropriate vectors $\vec{x}_i$.

- $\mathsf{RP}_{\mathsf{Lat}}.\mathsf{Init}(\vec{c} = \boldsymbol{A} \cdot \vec{r} + (\vec{0} \parallel \vec{x}), \vec{x} \in [0, \vec{B}], \vec{r} \xleftarrow{\$} D_\sigma^l)$:
  1. compute $\vec{x}_i$ s.t. $4\vec{x}(\vec{B} - \vec{x}) + \vec{1} = \sum_{i=1..3} \vec{x}_i^2$
  2. Set $\vec{r}_0 = -\vec{r}, \vec{x}_0 = \vec{B} - \vec{x}$ and $\vec{c}_0 = (\vec{0} \parallel \vec{B}) - \vec{c}$
  3. Set $\forall i \in [0, 3] : \vec{m}_i \xleftarrow{\$} [\vec{0}, \vec{B}CL]$
  4. Set $\vec{x}_4 = 4 \cdot \vec{m}_0(\vec{B} - 2\vec{x}_0) - 2 \cdot \sum_{i=1..3} \vec{x}_i \vec{m}_i$ and $\vec{m}_4 = -4(\vec{m}_0)^2 - \sum_{i=1..3} (\vec{m}_i)^2$
  5. Set $\forall i \in [1, 4] : \vec{r}_i \xleftarrow{\$} D_\sigma^l$
  6. Set $\forall i \in [0, 4] : \vec{s}_i \xleftarrow{\$} D_{\sigma_{\mathsf{rej}}}^l$
  7. Set $\forall i \in [1, 4] : \vec{c}_i = \boldsymbol{A} \cdot \vec{r}_i + (\vec{0} \parallel \vec{x}_i)$
  8. Set $\forall i \in [0, 4] : \vec{d}_i = \boldsymbol{A} \cdot \vec{s}_i + (\vec{0} \parallel \vec{m}_i)$
  9. Output $\{\vec{c}_i\}_{i=1..4}, \{\vec{d}_i\}_{i=0..4}$
- $\mathsf{RP}_{\mathsf{Lat}}.\mathsf{Chall}()$: Output $\gamma \xleftarrow{\$} [-C, C]$
- $\mathsf{RP}_{\mathsf{Lat}}.\mathsf{Resp}(\gamma)$:
  1. Sets $\forall i \in [0, 3] : \vec{z}_i = \vec{m}_i + \gamma \cdot \vec{x}_i$
  2. Set $\forall i \in [0, 4] : \vec{t}_i = \vec{s}_i + \gamma \cdot \vec{r}_i$
  3. For all $i \in [0, 4]$: abort with probability $1 - \min\left(\frac{D_{\sigma_{\mathsf{rej}}}^l(\vec{t}_i)}{M \cdot D_{\gamma\vec{r}_i, \sigma_{\mathsf{rej}}}^{l_2+n+l_1}(\vec{t}_i)}, 1\right)$
  4. Outputs $\{\vec{z}_i\}_{i=0..3}, \{\vec{t}_i\}_{i=0..4}$
- $\mathsf{RP}_{\mathsf{Lat}}.\mathsf{Verify}(\{\vec{c}_i\}_{i=1..4}, \{\vec{d}_i\}_{i=0..4}, \gamma, \{\vec{z}_i\}_{i=0..3}, \{\vec{t}_i\}_{i=0..4})$:
  1. Check $\forall i \in [0, 4] : \|\vec{t}_i\| \leq 2\sqrt{l} \cdot (\sigma_{\mathsf{rej}} + C \cdot \sigma)$
  2. Check $\forall i \in [1, 3] : \vec{z}_i \in [\vec{0}, \vec{B}CL]$
  3. Set $\vec{c}_0 = (\vec{0} \parallel \vec{B}) - \vec{c}$
  4. Set $\vec{z}_4 = 4 \cdot \vec{z}_0(\gamma\vec{B} - \vec{z}_0) + \vec{\gamma}^2 - \sum_{i=1..3} \vec{z}_i^2$
  5. Check $\forall i \in [0, 4] : \boldsymbol{A} \cdot \vec{t}_i + (\vec{0} \parallel \vec{z}_i) = \vec{d}_i + \gamma \cdot \vec{c}_i$

Note that we check $\vec{z}_i \in [B\vec{C}L]$ instead of $\vec{z}_i \in [BC(\vec{L} + 1)]$, since rejection sampling already brings non-perfect completeness with certain parameter choices. This introduces an additional completeness error of $4/L$.

Theorem 4. *The scheme* $\mathsf{RP}_{\mathsf{Lat}}$ *is correct with correctness error* $1-(1/M)^5+10(2^l e^{-3l/2})+4/L$.

Proof. The three-square decomposition $\{x_i\}_{i=1..3}$ can be efficiently computed by lemma 1. We show that each check of the verifier is confirmed if both parties are honest. By definition of σ and lemma 11, the algorithm outputs something with probability $1-(1/M)^5$. For all $i \in [0,4]$, we have $\|\vec{r}_i\| \leq 2\sigma\sqrt{l}$ with probability $1-5(2^l e^{-3l/2})$ and thus $\|\vec{t}_i\| \leq 2\sigma_{\mathsf{rej}}\sqrt{l}+2C\sigma\sqrt{l} = 2\sqrt{l}\cdot(\sigma_{\mathsf{rej}}+C\sigma)$ with the same probability by lemma 10. Also, for $i \in [0,3]$ it holds that:

$$\begin{aligned}\boldsymbol{A}\cdot\vec{t}_i+(\vec{0}^n \parallel \vec{z}_i) &= \boldsymbol{A}\cdot(\vec{s}_i+\gamma\cdot\vec{r}_i)+(\vec{0}\parallel\vec{m}_i+\gamma\cdot\vec{x}_i)\\ &= (\boldsymbol{A}\cdot\vec{s}_i+(\vec{0}^n\parallel\vec{m}_i))+\gamma(\boldsymbol{A}\cdot\vec{r}_i+(\vec{0}\parallel\vec{x}_i))\\ &= \vec{d}_i+\gamma\cdot\vec{c}_i\end{aligned}$$

By lemma 8, it holds that $\vec{z}_i \in [0, B\vec{C}L]$ for $i \in [1,3]$ with probability $1/L$ and because $4\vec{x}(\vec{B}-\vec{x})+\vec{1} = \sum_{i=1..3}\vec{x}_i^2$ holds by construction, we have:

$$\begin{aligned}\boldsymbol{A}\cdot\vec{t}_4+(\vec{0}\parallel\vec{z}_4) &= \boldsymbol{A}\cdot(\gamma\cdot\vec{r}_4+\vec{s}_4)+(\vec{0}\parallel 4\cdot\vec{z}_0(\gamma\vec{B}-\vec{z}_0)+\vec{\gamma}^2-\sum_{i=1..3}\vec{z}_i^2)\\ &= (\boldsymbol{A}\cdot\vec{s}_4)+(\vec{0}\parallel\vec{m}_4))+\gamma\cdot(\boldsymbol{A}\cdot\vec{r}_4+(\vec{0}\parallel\vec{z}_4))\\ &= \vec{d}_4+\gamma\cdot\vec{c}_4\end{aligned}$$

Theorem 5. *Suppose* $L \geq 16$. *Then* $\mathsf{RP}_{\mathsf{Lat}}$ *for* $[0,B]$ *satisfies 3-special soundness with soundness error* $\frac{2}{2C+1}$ *under the* $\mathsf{SIS}_{l_1,l,q,4CT}$ *assumption (and additionally the* CRHF *assumption if a hash function si used for compressing the first flow).*

Proof. Assume that a PPT adversary can interactively produce three valid transcripts with challenges γ,γ',γ'' and witnesses $[\vec{t}_i,\vec{z}_i,\vec{t}_4]$, $[\vec{t}_i',\vec{z}_i',\vec{t}_4']$ and $[\vec{t}_i'',\vec{z}_i'',\vec{t}_4'']$ for $i \in [0,3]$. We denote by $\overline{a},\underline{a}$ the difference of $a'-a, a''-a$ respectively for $a \in \{\vec{t}_i,\vec{z}_i,\vec{t}_y\}$. Also, we assume without loss of generality that $\overline{\gamma}>0, \underline{\gamma}>0$. First, for all $i \in [0,4]$ it holds that:

$$\begin{aligned}&\boldsymbol{A}\cdot\vec{t}_i+(\vec{0}\parallel\vec{z}_i)=\vec{d}_i+\gamma\cdot\vec{c}_i \wedge \boldsymbol{A}\cdot\vec{t}_i'+(\vec{0}\parallel\vec{z}_i')=\vec{d}_i+\gamma'\cdot\vec{c}_i\\ &\wedge\ \boldsymbol{A}\cdot\vec{t}_i''+(\vec{0}\parallel\vec{z}_i'')=\vec{d}_i+\gamma''\cdot\vec{c}_i\\ \Longrightarrow\ &\boldsymbol{A}\cdot\overline{\vec{t}_i}+(\vec{0}\parallel\overline{\vec{z}_i})=\overline{\gamma}\cdot\vec{c}_i\wedge\boldsymbol{A}\cdot\underline{\vec{t}_i}+(\vec{0}\parallel\underline{\vec{z}_i})=\underline{\gamma}\cdot\vec{c}_i\\ \Longrightarrow\ &\boldsymbol{A}\cdot\overline{\vec{t}_i}+\overline{\gamma}(\vec{0}\parallel\overline{\vec{z}_i}/\overline{\gamma})=\overline{\gamma}\cdot\vec{c}_i\wedge\boldsymbol{A}\cdot\underline{\vec{t}_i}+\underline{\gamma}(\vec{0}\parallel\underline{\vec{z}_i}/\underline{\gamma})=\underline{\gamma}\cdot\vec{c}_i\end{aligned}$$

Additionally, we have $\|\overline{\vec{t}_i}\| \leq \|\vec{t}_i'\|+\|\vec{t}_i\| \leq 4\sqrt{l}\cdot(\sigma_{\mathsf{rej}}+C\cdot\sigma)$ and equally $\left\|\underline{\vec{t}_i}\right\|_2 \leq 4\sqrt{l}\cdot(\sigma_{\mathsf{rej}}+C\cdot\sigma)$. Thus for $i \in [1,4]$, the commitments $\vec{c}_i$ commits to message $\vec{x}_i = \overline{\vec{z}_i}/\overline{\gamma}$ with opening $(\overline{\vec{t}_i},\overline{\gamma})$ and because of the binding property of Lat also $\vec{x}_i = \underline{\vec{z}_i}/\underline{\gamma}$.

Similarly, we set $\vec{x}_0 = \overline{\vec{z}_0}/\overline{\gamma} = \underline{\vec{z}_0}/\underline{\gamma}$ which would be committed in $\vec{c}_0$ if it were interpreted as Lat commitment. This shows that $\vec{c}$ is a commitment to message $\vec{x} = \lfloor(\overline{\gamma}\cdot\vec{B}-\overline{\vec{z}_0})/\overline{\gamma}\rfloor = \vec{B}-\lfloor\overline{\vec{z}_0}/\overline{\gamma}\rfloor$ with opening $(\overline{\vec{t}_i},\overline{\gamma},\overline{\gamma}\cdot\vec{B}-\overline{\vec{z}_0})$. Note that the openings are small enough, i.e.

$$\begin{aligned}\left\|\overline{\gamma}\cdot\vec{B}-\overline{\vec{z}_0}\right\|_\infty &\leq 2CB+BCL\\ &\leq 3BCL\\ &\leq U/C,\\ |\gamma| &\leq C.\end{aligned}$$

Because of the binding property of $\overline{\mathsf{Lat}}$ also $\vec{x} = \vec{B} - \lfloor \underline{\vec{z_0}}/\underline{\gamma} \rfloor$. All that is left to check is that $\vec{x}$ is indeed in the desired bounds. Denoting $\vec{\mu}_i = \vec{z}_i - \gamma \cdot \vec{x}_i$ for $i \in [0,3]$, we have:

$$\begin{aligned}
\vec{z}'_i - \vec{\mu}_i &= \vec{z}'_i - \vec{z}_i + \gamma \cdot \vec{x}_i \\
&= \overline{\vec{z}_i} + \gamma \cdot \vec{x}_i \\
&= \overline{\gamma} \cdot \vec{x}_i + \gamma \cdot \vec{x}_i \\
&= (\gamma' - \gamma) \cdot \vec{x}_i + \gamma \cdot \vec{x}_i \\
&= \gamma' \cdot \vec{x}_i, \\
\vec{z}''_i - \vec{\mu}_i &= \vec{z}''_i - \vec{z}_i + \gamma \cdot \vec{x}_i \\
&= \underline{\vec{z}_i} + \gamma \cdot \vec{x}_i \\
&= \underline{\gamma} \cdot \vec{x}_i + \gamma \cdot \vec{x}_i \\
&= (\gamma'' - \gamma) \cdot \vec{x}_i + \gamma \cdot \vec{x}_i \\
&= \gamma'' \cdot \vec{x}_i.
\end{aligned}$$

And with that, we can identify vectors $\vec{x}_i$ in $\vec{z}_4$:

$$\begin{aligned}
\vec{z}_4 &= 4 \cdot \vec{z}_0(\gamma\vec{B} - \vec{z}_0) + \vec{\gamma}^2 - \sum_{i=1..3} \vec{z}_i^2 \\
&= 4 \cdot (\gamma\vec{x}_0 + \vec{\mu}_0)(\gamma\vec{B} - (\gamma\vec{x}_0 + \vec{\mu}_0)) + \vec{\gamma}^2 - \sum_{i=1..3} (\gamma\vec{x}_i + \vec{\mu}_i)^2 \\
&= 4[\gamma\vec{x}_0(\gamma\vec{B} - \gamma\vec{x}_0 - \gamma\vec{\mu_0}(\vec{B} - 2\vec{x}_0) - \vec{\mu_0}^2] + \vec{\gamma}^2 \\
&- \sum_{i=1..3} [(\gamma\vec{x}_i)^2 + 2c\vec{x}_i\vec{\mu}_i + \vec{\mu}_i^2] \\
&= \gamma^2[4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} - \sum_{i=1..3} \vec{x}_i^2] \\
&+ \gamma[4\vec{\mu_0}(\vec{B} - 2\vec{x}_0) - 2 \sum_{i=1..3} \vec{x}_i\vec{\mu}_i] \\
&- 4\vec{\mu}_0^2 - \sum_{i=0..3} \vec{\mu}_i^2
\end{aligned}$$

Setting $\vec{\phi} = 4\vec{\mu_0}(\vec{B} - 2\vec{x}_0) - 2\sum_{i=1..3} \vec{x}_i\vec{\mu}_i$ and $\vec{\psi} = -4\vec{\mu}_0^2 - \sum_{i=0..3} \vec{\mu}_i^2$, we equally obtain:

$$\begin{aligned}
\vec{z}_4 &= \gamma^2[4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} - \sum_{i=1..3} \vec{x}_i^2] + \gamma\vec{\phi} + \vec{\psi} \\
\vec{z}'_4 &= (\gamma')^2[4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} - \sum_{i=1..3} \vec{x}_i^2] + \gamma'\vec{\phi} + \vec{\psi} \\
\vec{z}''_4 &= (\gamma'')^2[4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} - \sum_{i=1..3} \vec{x}_i^2] + \gamma''\vec{\phi} + \vec{\psi}
\end{aligned}$$

Further since $\overline{\vec{z}_4}/\overline{\gamma} = \underline{\vec{z}_4}/\underline{\gamma}$, we obtain:

$$
\begin{aligned}
&\vec{z}'_4 - \vec{z}_4 = ((\gamma')^2 - \gamma^2)[4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} - \sum_{i=1..3} \vec{x}_i^2] + (\gamma' - \gamma)\vec{\phi} \wedge \\
&\vec{z}''_4 - \vec{z}_4 = ((\gamma'')^2 - \gamma^2)[4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} - \sum_{i=1..3} \vec{x}_i^2] + (\gamma'' - \gamma)\vec{\phi} \\
\implies &\overline{\vec{z}_4}/\overline{\gamma} = (\gamma' + \gamma)[4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} - \sum_{i=1..3} \vec{x}_i^2] + \vec{\phi} \wedge \\
&\underline{\vec{z}_4}/\underline{\gamma} = (\gamma'' + \gamma)[4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} - \sum_{i=1..3} \vec{x}_i^2] + \vec{\phi} \\
\implies &(\gamma' + \gamma)[4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} - \sum_{i=1..3} \vec{x}_i^2] = \\
&(\gamma'' + \gamma)[4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} - \sum_{i=1..3} \vec{x}_i^2] \\
\implies &(\gamma' - \gamma'')[4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} - \sum_{i=1..3} \vec{x}_i^2] = 0 \\
\implies &4\vec{x}_0(\vec{B} - \vec{x}_0) + \vec{1} = \sum_{i=1..3} \vec{x}_i^2 \\
\implies &4\overline{\vec{z}_0}(\overline{\gamma}\vec{B} - \overline{\vec{z}_0}) + \overline{\vec{\gamma}}^2 = \sum_{i=1..3} \overline{\vec{z}_i}^2
\end{aligned}
$$

This implies that $4\overline{\vec{z}_0}(\overline{\gamma}\vec{B} - \overline{\vec{z}_0}) + \overline{\vec{\gamma}}^2 \geq \vec{0}$ in $\mathbb{Q}^n$ and further that $\vec{x} \in [0, \vec{B}]$. In more detail, we show that both sides of the equation are short which implies that it holds over the integers. We set $K = (BCL)$. For the right side, it holds that $\|\sum_{i=1..3} \overline{\vec{z}_i}^2\|_\infty \leq 3 \cdot (2BCL)^2 = 12K^2 \leq U \leq \frac{q-1}{2}$. Similarly, $\|4\overline{\vec{z}_0}(\overline{\gamma}\vec{B} - \overline{\vec{z}_0}) + \overline{\vec{\gamma}}^2\|_\infty \leq 8K^2 + 16K^2 + K^2 \leq 32K^2 \leq U$. Thus $\overline{\vec{z}_0}(\overline{\gamma}\vec{B} - \overline{\vec{z}_0}) + \overline{\vec{\gamma}}^2 \geq \vec{0}$ in $\mathbb{Q}^n$ and lemma 13 yields that $x = \lfloor \overline{\vec{z}}/\overline{\gamma} \rceil \in [0, \vec{B}]$

Theorem 6. *The scheme* $\mathsf{RP}_{\mathsf{Lat}}$ *is HVZK under the* $\mathsf{LWE}_{l_2,l,q,\sigma}$ *with simulation error* $4/L$.

Proof. We show that the scheme is zero-knowledge by modifying an honest transcript, given the challenge γ, in 4 indistinguishable steps into a transcript that does not require the witnesses.

Game 1:
Outputs an unmodified transcript from an interaction of an honest verifier and prover from the definition:

- compute $\vec{x}_i$ s.t. $4\vec{x}(\vec{B} - \vec{x}) + \vec{1} = \sum_{i=1..3} \vec{x}_i^2$
- Set $\vec{r}_0 = -\vec{r}, \vec{x}_0 = \vec{B} - \vec{x}$ and $\vec{c}_0 = (\vec{0} \parallel \vec{B}) - \vec{c}$
- Set $\forall i \in [0,3] : \vec{m}_i \xleftarrow{\$} [\vec{0}, B\vec{C}L]$
- Set $\vec{z}_4 = 4 \cdot \vec{m}_0(\vec{B} - 2\vec{x}_0) - 2 \cdot \sum_{i=1..3} \vec{x}_i\vec{m}_i$ and $\vec{m}_4 = -4(\vec{m}_0)^2 - \sum_{i=1..3}(\vec{m}_i)^2$
- Set $\forall i \in [1,4] : \vec{r}_i \xleftarrow{\$} D_\sigma^l$
- Set $\forall i \in [0,4] : \vec{s}_i \xleftarrow{\$} D_{\sigma_{\mathsf{rej}}}^l$
- Set $\forall i \in [1,4] : \vec{c}_i = \boldsymbol{A} \cdot \vec{r}_i + (\vec{0} \parallel \vec{x}_i)$
- Set $\forall i \in [0,4] : \vec{d}_i = \boldsymbol{A} \cdot \vec{s}_i + (\vec{0} \parallel \vec{m}_i)$
- set $\forall i \in [0,3] : \vec{z}_i = \vec{m}_i + \gamma \cdot \vec{x}_i$
- set $\forall i \in [0,4] : \vec{t}_i = \vec{s}_i + \gamma \cdot \vec{r}_i$
- for all $i \in [0,4]$: abort with probability $1 - \min\left(\frac{D^l_{\sigma_{\mathsf{rej}}}(\vec{t}_i)}{M \cdot D^{l_2+n+l_1}_{\gamma\vec{r}_i,\sigma_{\mathsf{rej}}}(\vec{t}_i)}, 1\right)$
- outputs $\{\{\vec{c}_i\}_{i=1..4}, \{\vec{d}_i\}_{i=0..4}, \{\vec{z}_i\}_{i=0..3}, \{\vec{t}_i\}_{i=0..4}\}$

for given $\gamma \in [-C, C], \vec{c} \leftarrow \overline{\mathsf{Lat}}.\mathsf{Commit}(\vec{x}; \vec{r}), \vec{x}, \vec{r}$.

Game 2:
Rewrites the mask commitments as in the verification check:

- compute $\vec{x}_i$ s.t. $4\vec{x}(\vec{B}-\vec{x})+\vec{1}=\sum_{i=1..3}\vec{x}_i^2$
- Set $\vec{r}_0=-\vec{r}, \vec{x}_0=\vec{B}-\vec{x}$ and $\vec{c}_0=(\vec{0}\parallel\vec{B})-\vec{c}$
- Set $\forall i\in[0,3]:\vec{m}_i\xleftarrow{\$}[\vec{0},B\vec{C}L]$
- Set $\vec{z}_4=4\cdot\vec{m}_0(\vec{B}-2\vec{x}_0)-2\cdot\sum_{i=1..3}\vec{x}_i\vec{m}_i$ and $\vec{m}_4=-4(\vec{m}_0)^2-\sum_{i=1..3}(\vec{m}_i)^2$
- Set $\forall i\in[1,4]:\vec{r}_i\xleftarrow{\$}D^l_\sigma$
- Set $\forall i\in[0,4]:\vec{s}_i\xleftarrow{\$}D^l_{\sigma_{\mathsf{rej}}}$
- set $\forall i\in[0,3]:\vec{z}_i=\vec{m}_i+\gamma\cdot\vec{x}_i$
- set $\forall i\in[0,4]:\vec{t}_i=\vec{s}_i+\gamma\cdot\vec{r}_i$
- Set $\boxed{\vec{z}_4=4\cdot\vec{z}_0(\gamma\vec{B}-\vec{z}_0)+\vec{\gamma}^2-\sum_{i=1..3}\vec{z}_i^2}$
- Set $\forall i\in[1,4]:\vec{c}_i=\boldsymbol{A}\cdot\vec{r}_i+(\vec{0}\parallel\vec{x}_i)$
- set $\forall i\in[0,4]:\boxed{\vec{d}_i=\boldsymbol{A}\cdot\vec{t}_i+(\vec{0}^n\parallel\vec{z}_i)-\gamma\cdot\vec{c}_i}$
- for all $i\in[0,4]$: abort with probability $1-\min\left(\frac{D^l_{\sigma_{\mathsf{rej}}}(\vec{t}_i)}{M\cdot D^{l_2+n+l_1}_{\gamma\vec{r}_i,\sigma_{\mathsf{rej}}}(\vec{t}_i)},1\right)$
- outputs $\{\{\vec{c}_i\}_{i=1..4},\{\vec{d}_i\}_{i=0..4},\{\vec{z}_i\}_{i=0..3},\{\vec{t}_i\}_{i=0..4}\}$

for given $\gamma\in[-C,C],\vec{c}\leftarrow\overline{\mathsf{Lat}}.\mathsf{Commit}(\vec{x};\vec{r}),\vec{x},\vec{r}$.

Claim. Game 1 and game 2 are identify distributed.

Proof. The equalities are shown in the correctness proof.

Game 3:
Removes the dependency on the witness $\vec{r}_i$ in the zero-knowledge witnesses $\vec{t}_i$ for $i\in[0,4]$:

- compute $\vec{x}_i$ s.t. $4\vec{x}(\vec{B}-\vec{x})+\vec{1}=\sum_{i=1..3}\vec{x}_i^2$
- Set $\vec{r}_0=-\vec{r}, \vec{x}_0=\vec{B}-\vec{x}$ and $\vec{c}_0=(\vec{0}\parallel\vec{B})-\vec{c}$
- Set $\forall i\in[0,3]:\vec{m}_i\xleftarrow{\$}[\vec{0},B\vec{C}L]$
- Set $\vec{z}_4=4\cdot\vec{m}_0(\vec{B}-2\vec{x}_0)-2\cdot\sum_{i=1..3}\vec{x}_i\vec{m}_i$ and $\vec{m}_4=-4(\vec{m}_0)^2-\sum_{i=1..3}(\vec{m}_i)^2$
- set $\forall i\in[0,4]:\boxed{\vec{t}_i\xleftarrow{\$}D^l_{\sigma_{\mathsf{rej}}}}$
- Set $\forall i\in[1,4]:\vec{r}_i\xleftarrow{\$}D^l_\sigma$
- set $\forall i\in[0,3]:\vec{z}_i=\vec{m}_i+\gamma\cdot\vec{x}_i$
- Set $\forall i\in[1,4]:\vec{c}_i=\boldsymbol{A}\cdot\vec{r}_i+(\vec{0}\parallel\vec{x}_i)$
- set $\forall i\in[0,4]:\vec{d}_i=\boldsymbol{A}\cdot\vec{t}_i+(\vec{0}^n\parallel\vec{z}_i)-\gamma\cdot\vec{c}_i$
- for all $i\in[0,4]$: abort with probability $\boxed{1-1/M}$
- outputs $\{\{\vec{c}_i\}_{i=1..4},\{\vec{d}_i\}_{i=0..4},\{\vec{z}_i\}_{i=0..3},\{\vec{t}_i\}_{i=0..4}\}$

for given $\gamma\in[-C,C],\vec{c}\leftarrow\overline{\mathsf{Lat}}.\mathsf{Commit}(\vec{x};\vec{r}),\vec{x},\vec{r}$.

Claim. Game 2 and game 3 are statistically indistinguishable.

Proof. We define subgames 2.i for $i\in[-1,4]$, where subgame 2.(-1) is identical to game 2 and subgame 2.i is identical to subgame 2.$(i-1)$ for $i\in[0,4]$ but the vector $\vec{t}_i$ is sampled from $D^l_{\sigma_{\mathsf{rej}}}$ and in the abortion check, the subgame aborts with probability $1-1/M$. Now, the adjacent subgames only differ in the form of a rejection sampling simulation simulation as in lemma 11 as shown in the following.

Let $i\in[0,4]$ and let $\mathscr{A}$ be an adversary that can distinguish subgame 2.(i-1) and subgame 2.i. We construct an adversary $\mathscr{B}$ distinguishing between the algorithms in lemma 11. Let h be the

distribution of $\vec{v} = \gamma \cdot \vec{r}_i$, i.e. $\gamma \in [-C, C], \vec{r}_i \xleftarrow{\$} D^l_\sigma$. Then $\|\vec{v}\| \leq T := 2p\sigma\sqrt{l_2 + n + l_1}$ by lemma 10 and thus $\sigma_{\mathsf{rej}} = \omega(T\sqrt{\log l_2 + n + l_1})$. Let $(\vec{t}, \vec{v}), \mathit{aborted} \in \{0, 1\}$ be the output of either algorithm $\mathscr{T}$ or $\mathscr{S}$ of lemma 11, i.e. we assume if the algorithm aborts it outputs $\mathit{aborted} = 1$ and $(\vec{t}, \vec{v})$ is invalid. The distinguisher $\mathscr{B}$ proceeds as in subgame 2.3 but it aborts if $\mathit{aborted} = 1$ and it instead sets $\vec{t}_i = \vec{t}$ and $\vec{r}_i = \vec{v} \cdot \gamma^{-1}$ which is possible if $\gamma \neq 0$. Note that if $\gamma = 0$ the games are identically distributed if not-aborted and thus indistinguishable since the abortion probability is negligible.

Now, if $(\vec{t}, \vec{v}), \mathit{aborted}$ is sampled as in $\mathscr{T}$, the distribution is identical to subgame 2.3 and otherwise the distribution is identical to subgame 2.4. This concludes the claim.

Game 4:
Removes the dependency on the witness $\vec{x}_i$ in the zero-knowledge witnesses $\vec{z}_i$ for $i \in [0, 3]$:

- compute $\vec{x}_i$ s.t. $4\vec{x}(\vec{B} - \vec{x}) + \vec{1} = \sum_{i=1..3} \vec{x}_i^2$
- Set $\vec{r}_0 = -\vec{r}, \vec{x}_0 = \vec{B} - \vec{x}$ and $\vec{c}_0 = (\vec{0} \parallel \vec{B}) - \vec{c}$
- set $\forall i \in [0, 3] : \boxed{\vec{z}_i \xleftarrow{\$} [\vec{0}, B\vec{C}L]}$
- Set $\forall i \in [0, 3] : \boxed{\vec{m}_i = \vec{z}_i - \gamma \cdot \vec{x}_i}$
- Set $\vec{z}_4 = 4 \cdot \vec{m}_0(\vec{B} - 2\vec{x}_0) - 2 \cdot \sum_{i=1..3} \vec{x}_i \vec{m}_i$ and $\vec{m}_4 = -4(\vec{m}_0)^2 - \sum_{i=1..3} (\vec{m}_i)^2$
- set $\forall i \in [0, 4] : \vec{t}_i \xleftarrow{\$} D^l_{\sigma_{\mathsf{rej}}}$
- Set $\forall i \in [1, 4] : \vec{r}_i \xleftarrow{\$} D^l_\sigma$
- Set $\vec{z}_4 = 4 \cdot \vec{z}_0(\gamma\vec{B} - \vec{z}_0) + \vec{\gamma}^2 - \sum_{i=1..3} \vec{z}_i^2$
- Set $\forall i \in [1, 4] : \vec{c}_i = \boldsymbol{A} \cdot \vec{r}_i + (\vec{0} \parallel \vec{x}_i)$
- set $\forall i \in [0, 4] : \vec{d}_i = \boldsymbol{A} \cdot \vec{t}_i + (\vec{0}^n \parallel \vec{z}_i) - \gamma \cdot \vec{c}_i$
- for all $i \in [0, 4]$: abort with probability $1 - 1/M$
- outputs $\{\{\vec{c}_i\}_{i=1..4}, \{\vec{d}_i\}_{i=0..4}, \{\vec{z}_i\}_{i=0..3}, \{\vec{t}_i\}_{i=0..4}\}$

for given $\gamma \in [-C, C], \vec{c} \leftarrow \overline{\mathsf{Lat}}.\mathsf{Commit}(\vec{x}; \vec{r}), \vec{x}, \vec{r}$.

Claim. Game 3 and Game 4 are statistically indistinguishable.

Proof. The claim follows directly from lemma 8 applied on each coordinate and since $\vec{0} \leq \vec{x}_i \leq 2^{\vec{B}}$.

Game 5:
Removes the dependency on $\vec{x}_i$ in the commitment to $\vec{c}_i$:

- Set $\vec{c}_0 = (\vec{0} \parallel \vec{B}) - \vec{c}$
- set $\forall i \in [0, 3] : \vec{z}_i \xleftarrow{\$} [\vec{0}, B\vec{C}L]$
- set $\forall i \in [0, 4] : \vec{t}_i \xleftarrow{\$} D^l_{\sigma_{\mathsf{rej}}}$
- Set $\forall i \in [1, 4] : \vec{r}_i \xleftarrow{\$} D^l_\sigma$
- Set $\vec{z}_4 = 4 \cdot \vec{z}_0(\gamma\vec{B} - \vec{z}_0) + \vec{\gamma}^2 - \sum_{i=1..3} \vec{z}_i^2$
- Set $\forall i \in [1, 4] : \boxed{\vec{c}_i = \boldsymbol{A} \cdot \vec{r}_i + (\vec{0} \parallel \vec{0})}$
- set $\forall i \in [0, 4] : \vec{d}_i = \boldsymbol{A} \cdot \vec{t}_i + (\vec{0}^n \parallel \vec{z}_i) - \gamma \cdot \vec{c}_i$
- for all $i \in [0, 4]$: abort with probability $1 - 1/M$
- outputs $\{\{\vec{c}_i\}_{i=1..4}, \{\vec{d}_i\}_{i=0..4}, \{\vec{z}_i\}_{i=0..3}, \{\vec{t}_i\}_{i=0..4}\}$

for given $\gamma \in [-C, C], \vec{c} \leftarrow \overline{\mathsf{Lat}}.\mathsf{Commit}(\vec{x}; \vec{r}), \vec{x}, \vec{r}$.

Claim. Game 4 and Game 5 are indistinguishable under the hiding property of the commitment scheme Lat.

Proof. We define subgames 4.i for $i \in [0, 4]$, where subgame 4.0 is identical to game 4 and subgame 4.i is identical to subgame 4.(i-1) for $i \in [1, 4]$ but the the commitment $\vec{c}_i = \boldsymbol{A} \cdot \vec{r}_i + (\vec{0} \parallel \vec{0})$ commits to $\vec{0}$ instead of $\vec{x}_i$.

For $i \in [1,4]$, we provide an algorithm $\mathscr{B}$ breaking the hiding property of Lat using a distinguisher $\mathscr{A}$ of subgame 4.(i-1) and subgame 4.i. First, $\mathscr{B}$ sets up the parameters of the range proof with the $\mathsf{pp} = \boldsymbol{B}$ received from the hiding challenger $\mathscr{C}$. Then it proceeds as in game 4.(i-1), with the adaption of $\vec{c}_i = \vec{c}'$ for the received vector $\vec{c}'$ from $\mathscr{C}$ on input $m_0 = \vec{z}_4$ and $m_1 = \vec{0}$. Next, it sends the computed transcript to $\mathscr{A}$. After receiving b from $\mathscr{A}$ it forwards the bit to the challenger $\mathscr{C}$.

Now, if challenger committed to m_0, the output of $\mathscr{B}$ is distributed as in game subgame subgame 4.(i-1) and otherwise as in subgame 4.i. Thus if $\mathscr{A}$ has non-negligible advantage, $\mathscr{B}$ breaks the hiding property of the commitment scheme.

6.4 Optimizations

Rejection Sampling for $\vec{z}_i$. In the above protocol, we mask the committed values $\vec{x}_i$ with a mask $\vec{m}_i$ and set $\vec{z}_i = \vec{m}_i + \gamma \cdot \vec{x}_i$. The mask $\vec{m}_i$ is chosen such that $\gamma \cdot \vec{x}_i$ is statistically blinded and that the resulting masked witness is kept short. In general, we can use the rejection sampling technique (lemma 11) that is utilized for blinding the small randomness $\gamma \cdot \vec{r}_i$ to also blind $\gamma \cdot \vec{x}_i$. Note that $\vec{x}_i$ has length n and thus $M_x = e^{\frac{13.3}{\log(n)} + \frac{1}{2\log^2(n)}}$. For a low correctness error, M needs to be minimized. For example $n \geq 140$ results in a correctness error of at most 31% for all $\vec{x}_i$. If $n < 140$, we can group the vectors into $\vec{\chi} = \vec{x}_0 \parallel .. \parallel \vec{x}_4$ into a single one and blind this vector via rejection sampling. Thus, length $n' \geq 140$ for $n \geq 35$ and the correctness error is small enough in practice.

In the case of $n \geq 140$, the maximal size of the blinded vector is $\|\gamma \cdot \vec{x}_i\|_\infty \leq BC$. Thus $\|\gamma \cdot \vec{x}_i\|_2 \leq BC\sqrt{n}$ by lemma 9. According to lemma 10, we require $2^n e^{-\frac{3n}{2}} = \mathrm{negl}(\lambda)$ in order for the bound to apply with overwhelming probability. This already holds for $n \geq 109$ for $\lambda = 128$. Further, we require that $\sigma_x \geq nBC$, where σ_x is the standard deviation for the discrete Gaussian distribution for rejection sampling. Under these parameters, the overhead of the masking process is $L' = 2n^{3/2}$.

In the case of $35 \leq n < 140$, we group $\vec{x}_i$ in $\vec{\chi}$ and apply a single rejection sampling operation. As above, we obtain the masking overhead $L' = 2(n')^{3/2}$. For even lower batch sizes n, we can adapt the approach of the DLOG setting and sample each value separately with a sufficiently large α.

Since the maximal size of $\vec{z}_i$ determines the necessary size of the modulus and thus this technique reduces the size of the scheme further. Essentially, we can set $U = 32BCL'$.

Utilizing a Hash Function. In order to avoid sending the mask commitments $\vec{d}_i$, the prover can utilize a collision resistant hash function $\mathsf{H} : \{0,1\}^* \mapsto \{0,1\}^{2\lambda}$ to fix its choice in Init. Concretely, the scheme is adapted as follows:

- Instead of outputting $\{\vec{d}_i\}_{i=0..4}$ in Init, the prover sets $\Delta = \mathsf{H}(\{\vec{d}_i\}_{i=0..4})$,
- Instead of checking $\forall i \in [0,4] : \boldsymbol{A} \cdot \vec{t}_i + (\vec{0} \parallel \vec{z}_i) = \vec{d}_i + \gamma \cdot \vec{c}_i$ in Verify, the verifier sets $\vec{f}_i = \boldsymbol{A} \cdot \vec{t}_i + (\vec{0} \parallel \vec{z}_i) - \gamma \cdot \vec{c}_i$ and checks that $\Delta = \mathsf{H}(\{\vec{f}_i\}_{i=0..4})$ holds.

The collision resistance of H forces $\vec{f}_i = \vec{d}_i$ and because Δ was sent in Init, the prover can not cheat because he does not know the challenge yet. This technique was also used in section 5 to which we refer for details on the proof of security. Because the hash function has an image of only 2λ bits and sending 4 commitments requires $(l_1 + n)\log(q)$ bits, the size of the range proof is reduced by $(l_1 + n)(2\log(B) + 4\lambda + 6) - 2\lambda$ bit.

Non-Interactive Zero Knowledge Proof. Using the Fiat-Shamir transformation (definition 10), the scheme $\mathsf{RP}_{\mathsf{Lat}}$ or $\mathsf{RP}^*_{\mathsf{Lat}}$ can be transformed into a non-interactive range. This also removes the requirement of sending the challenge and thus saves C bits of communication.

Efficient Repetitions. If the value C^{-1} is not negligible, the protocol needs to be repeated for $N = \lceil \frac{\lambda}{\log(C)} \rceil$ rounds in order to achieve negligible soundness error. In repetitions, the values $\{\vec{x}_i\}_{i=1..3}$ were already computed and committed to in $\{\vec{c}_i\}_{i=1..3}$ and thus, the prover can avoid sending them again in the following repetitions. This saves $(N-1) \cdot 3(l_1 + n)\log(q)$ bits in the proof.

Table 2. Parameters and total proof size in MB of the optimized range proof.

	proof size	$\log(q)$		proof size	$\log(q)$
$(B = 2^{32}, \lambda = 80)$			$(B = 2^{64}, \lambda = 80)$		
n = 1000	1.55	126	n = 1000	2.57	190
n = 500	1.357	125	n = 500	2.289	189
n = 180	1.218	123	n = 180	2.09	187
$(B = 2^{32}, \lambda = 128)$			$(B = 2^{64}, \lambda = 128)$		
n = 1000	3.194	118	n = 1000	5.36	182
n = 500	2.824	117	n = 500	4.873	181
n = 180	2.611	115	n = 180	4.52	180

6.5 Efficiency

The entire proof consists of elements $\{\vec{c}_i\}_{i=1..4}, \{\vec{d}_i\}_{i=0..4}, \gamma, \{\vec{z}_i\}_{i=0..3}, \{\vec{t}_i\}_{i=0..4}$ and requires $N = \lceil \frac{\lambda}{\log(C)} \rceil$ repetitions. We compute concrete parameters, including the size of the proof, with the script provided in appendix C. The following modifications from section 6.4 were applied:

- Hash function to avoid sending the mask commitments,
- Avoid resending commitments $\vec{c}_i$ in repetitions,
- Rejection sampling to mask $\vec{x}_i$,

While choosing the parameters, we need to make sure that $\mathsf{SIS}_{l_1,l,q,4CT}$ and $\mathsf{LWE}_{l_2,l,q,D_\sigma}$ are hard to solve in practice. We adapt the approach of [Yan+19] for estimating the hardness of the SIS and LWE problem. In more detail, we use similar parameters for the commitment scheme and inspect the root Hermite factor (RHF) [GN08] for the respective lattice problems. To achieve 80 and 128 bit security, the corresponding RHF are 1.0048 and 1.0035 respectively. The RHF of $\mathsf{SIS}_{n,m,q,\beta}$ and $\mathsf{LWE}_{n,m,q,D_\sigma}$ are $2^{\frac{\log^2(\beta)}{4n\log(q)}}$ and $2^{\frac{\log^2(\alpha/5.31)}{4n\log(q)}}$ respectively for $\alpha = \sigma \cdot \sqrt{2\pi}/q$.

The results are summarized for a range $[0, B]$ of 32 and 64 bits in table 2. We choose $C = 2^{20}, C = 2^{16}$ for $\lambda = 80, \lambda = 128$ respectively. Further, we choose $(l_1 = 2655, l_2 = 2830), (l_1 = 3180, l_2 = 3620), (l_1 = 3120, l_2 = 3420), (l_1 = 3760, l_2 = 4510)$ for $(\lambda = 80, B = 2^{32}), (\lambda = 128, B = 2^{32}), (\lambda = 80, B = 2^{64}), (\lambda = 128, B = 2^{64})$ respectively.

Note that while [Yan+19] only considers a range of 1000 bits, this range is not very common in practical applications. We stress that even though our proof is amortized over a large number of range proofs for optimal use of space, the intervals defined by $\vec{B}$ do not need to be equal component-wise.

7 Unbounded Integer Commitments

In this section, we apply our technique to obtain range proofs in the class group setting.

7.1 Overview

The previously established commitment scheme of section 4.1 requires the parameters, especially the size of the underlying algebraic structure, to be scaled with the size of the committed integers. In this section, we establish a commitment scheme and an accompanying range proof in the class group setting without scaling parameters. The core difference to the $\mathbb{Z}_q$ setting is that class groups allow us to commit to *unbounded* dyadic rationals $\frac{a}{2^k}$ directly, and we have a canonical choice of representatives and absolute value for $\mathbb{Z}[1/2]$, unlike $\mathbb{Z}_q$, since there is no wrap-around to account for. Thus, the parameters of the commitment scheme do not need to be adapted to commit to larger integers, and our encoding (i.e. rounding) applies directly.

We use a variant of ElGamal encryption [ElG84] as our commitment scheme to which we apply our encoding technique. For the range proof, we use the standard Σ-protocol techniques for showing that the prover knows the committed value and that the square-decomposition holds. As before, our encoding then ensures that $x \in [0, B]$.

7.2 Parameters

Let $\mathbb{G}$ be a class group $\mathsf{Cl}(\Delta)$. Further, let U be a polynomial bounds with $\mathsf{ord}(\mathbb{G}) \leq 2^U$. (Beware that U here and U in section 5 are unrelated.) We set $S = 2^{U+\lambda}$. Lastly, let H be a collision resistant hash function.

7.3 Commitment Scheme

We define the integer commitment scheme CG over class group group $\mathbb{G} = \mathsf{Cl}(\Delta)$ with message space $\mathcal{M}_{\text{com}} = \mathbb{Z}$. Essentially, we apply the encoding technique to a class group variant of ElGamal encryption in order to deal with dyadic numbers. We emphasize that we explicitly include the random coins for sampling pp below. Since the other cases had setups which were invertible sampleable, it makes no difference there. However, this is not the case in the class group setting. The scheme is thus defined as follows:

- $\mathsf{CG.Setup}(1^\lambda)$: outputs $\mathsf{pp} = (\mathbb{G}, g, \rho_g, h, \rho_h)$ where $(g, \rho_g) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{G}), (h, \rho_h) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{G})$,
- $\mathsf{CG.Commit}_{\mathsf{pp}}(x)$: samples $r \leftarrow [0, S]$, computes $c_1 = g^r$, $c_2 = h^r \cdot g^x$, sets $c = (c_1, c_2), d = (x, 0, r)$ and outputs (c, d),
- $\mathsf{CG.Verify}_{\mathsf{pp}}((c_1, c_2), x, (y, \ell, r))$: verifies that the following checks pass:

$$c_1 = g^{\frac{r}{2^\ell}} \wedge c_2 = h^{\frac{r}{2^\ell}} \cdot g^{\frac{y}{2^\ell}} \wedge x = \lfloor \frac{y}{2^\ell} \rceil.$$

Note that the second verification check is possible because square roots are efficiently computable in class groups with imaginary order.

Lemma 14. *The commitment scheme* CG *is correct.*

Proof. For every $(g, \rho_g, h, \rho_h) \leftarrow \mathsf{CG.Setup}(1^\lambda), ((c_1, c_2), (x, 0, r)) \leftarrow \mathsf{com.Commit}(x)$, it holds that $c_1 = g^r, c_2 = h^r \cdot g^x$ and $x = \lfloor \frac{x}{1} \rceil$.

Lemma 15. *The commitment scheme* CG *is hiding under the* DXDH *assumption.*

Proof. First, by the DXDH assumption the game is indistinguishable from a hiding game where $c_2 = g^s \cdot g^x$ for some $s \xleftarrow{\$} [0, S]$. Note that g^s is indistinguishable from a random group element according to lemma 6. Thus, g^s blinds g^x statistically.

Lemma 16. *The commitment scheme* CG *is binding under the* ORD *assumption.*

Proof. Let $\mathscr{A}$ be a PPT adversary on the binding property of CG. We construct adversary $\mathscr{B}$ that breaks the ORD assumption using $\mathscr{A}$ as follows. $\mathscr{B}$ receives a description of $\mathbb{G}$ and runs $\mathsf{pp} = (\mathbb{G}, g, \rho_g, h, \rho_h) \leftarrow \mathsf{CG.Setup}(1^\lambda)$ and forwards pp to $\mathscr{A}$. Then, $\mathscr{A}$ produces commitment (c_1, c_2), messages $x_1, x_2 \in \mathbb{Z}$ with $x_1 \neq x_2$ and openings $(y_0, \ell_0, r_0), (y_1, \ell_1, r_1)$. Now, $\mathscr{B}$ checks whether $r_0/2^{\ell_0} \neq r_1/2^{\ell_1}$. If yes, $\mathscr{B}$ outputs the pair $(g, 2^{\ell_1} r_0 - 2^{\ell_0} r_1)$. Otherwise, $\mathscr{B}$ outputs the pair $(g, y_0 \cdot 2^{\ell_1} - y_1 \cdot 2^{\ell_0})$.

We now show that the success of $\mathscr{B}$ is equal to the success of $\mathscr{A}$. We assume w.l.o.g. that $g \neq 1$ and that $\mathscr{A}$ is successful. If $r_0/2^{\ell_0} \neq r_1/2^{\ell_1}$, then the output of $\mathscr{B}$ is a non-trivial ORD solution, as $g^{2^{\ell_1} r_0 - 2^{\ell_0} r_1} = 1$ due to $g^{\frac{r_0}{2^{\ell_0}}} = c_1 = g^{\frac{r_1}{2^{\ell_1}}}$. Otherwise, $r_0/2^{\ell_0} = r_1/2^{\ell_1}$ and because of $h^{\frac{r_0}{2^{\ell_0}}} \cdot g^{\frac{y_0}{2^{\ell_0}}} = c_2 = h^{\frac{r_1}{2^{\ell_1}}} \cdot g^{\frac{y_1}{2^{\ell_1}}}$, we have $g^{\frac{y_0}{2^{\ell_0}}} = g^{\frac{y_1}{2^{\ell_1}}}$. As $x_1 \neq x_0$, this implies that $(g, y_0 \cdot 2^{\ell_1} - y_1 \cdot 2^{\ell_0})$ is a non-trivial ORD solution.

7.4 Properties

As ElGamal commitments are additively homomorphic, this scheme has the same homomorphic properties as the bounded integer commitment scheme from section 4.1, without the risk of wrap-around. Essentially, it allows for multiplication and addition of constants and retains a limited form of an additive homomorphism, i.e. for commitment c_0, c_1 to x_0, x_1 the resulting commitment $c_0 \cdot c_1$ will either commit to $x_0 + x_1$ or $x_0 + x_1 + 1$. Note that in case the commitments are generated honestly, the first case will hold.

Further, for a given security parameter, the parameters do not need to be scaled to the size of the committed integers while allowing for an untrusted setup. Also note that in the previous version, we required C to be $\mathsf{poly}(\lambda)$. This restriction was needed for a statistical argument in the old soundness proof. With the changes to the underlying commitment scheme, this argument is not required anymore.

7.5 Range Proof

For $(g, \rho_g) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{G}), (h, \rho_h) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{G})$, the common reference string $\mathsf{crs}_{\mathsf{RP_{CG}}}$ of $\mathsf{RP_{CG}}$ is defined as (g, ρ_g, h, ρ_h). For consistency with DLOG, we use B, C, L and S. As noted before, we let $S = 2^{U+\lambda}$. Since the group order in unknown, we have no restrictions on B, C, L, and choose $L = 2^\lambda$ for strong security guarantees. Also, we set $C = 2^\lambda$.

- $\mathsf{RP_{CG}.Init}(c_{0,1} = g^{r_0}, c_{0,2} = h^{r_0} \cdot g^{x_0}, x_0 \in [0, B], r_0 \in [0, S])$:
  1. compute x_i s.t. $4x_0(B - x_0) + 1 = \sum_{i=1}^{3} x_i^2$
  2. Set $\forall i \in [1, 3] : r_i \xleftarrow{\$} [0, S], c_{i,1} = g^{r_i}, c_{i,2} = h^{r_i} g^{x_i}$
  3. Set $\forall i \in [0, 3] : m_i \xleftarrow{\$} [0, BCL], s_i \xleftarrow{\$} [0, CLS], d_{i,1} = g^{s_i}, d_{i,2} = h^{s_i} g^{m_i}$
  4. Set $\sigma \xleftarrow{\$} [0, 4BCLS], d = h^\sigma c_{0,2}^{-4m_0} \prod_{i=1..3} c_{i,2}^{-m_i}$
  5. Set $\Delta = H(\{d_{i,1}, d_{i,2}\}_{i=0..3}, d)$
  6. Outputs $\{c_i\}_{i=1..3}, \Delta$
- $\mathsf{RP_{CG}.Chall}()$: outputs $\gamma \xleftarrow{\$} [0, C]$.
- $\mathsf{RP_{CG}.Resp}(\gamma)$:
  1. Sets $\forall i \in [0, 3] : z_i = m_i + \gamma \cdot x_i, t_i = s_i + \gamma \cdot r_i$
  2. Sets $\tau = \sigma + \gamma(\sum_{i=1..3} x_i r_i + 4(B - x_0) r_0)$
  3. Outputs $\{z_i, t_i\}_{i=0..3}, \tau$
- $\mathsf{RP_{CG}.Verify}(\{c_{i,1}, c_{i,2}\}_{i=1..3}, \Delta, \gamma, \{z_i, t_i\}_{i=0..3}, \tau)$:
  1. Compute $\forall i \in [0, 3] : f_{i,1} = g^{t_i} c_{i,1}^{-\gamma}, f_{i,2} = g^{z_i} h^{t_i} c_{i,2}^{-\gamma}$
  2. Compute $f = h^\tau \cdot g^\gamma \cdot c_{0,2}^{4(\gamma B - z_0)} \cdot \prod_{i=1..3} c_{i,2}^{-z_i}$
  3. Check $\Delta = H(\{f_{i,1}, f_{i,2}\}_{i=0..3}, f)$
  4. Check $z_i \in [0, BC(L + 1)]$

Theorem 7. *The scheme* $\mathsf{RP_{CG}}$ *is correct.*

Proof. This follows by inspection.

Theorem 8. *The scheme* $\mathsf{RP_{CG}}$ *is* 2*-special sound under the* ORD*,* 2-fROOT *and* CRHF *assumptions with soundness error* $1/(C + 1)$*.*

Proof. We assume we have two accepting transcripts for distinct challenges $\gamma \neq \tilde{\gamma}$ with witnesses z_i, t_i, τ and $\tilde{z_i}, \tilde{t_i}, \tilde{\tau}$ respectively. Without loss of generality, say $\gamma > \tilde{\gamma}$. First, by the collision resistance of the hash function H, we have $d = f = \tilde{f}$ and $\forall i \in [0, 3], j \in [1, 2] : d_i = f_{i,j} = \tilde{f_{i,j}}$. We denote by $\overline{a}$ the difference of $a - \tilde{a}$ for $a \in \{z_i, t_i, \tau\}$.

From the first check, we obtain the following equations.

$$c_{i,1}^{\overline{\gamma}} = g^{\overline{t_i}} \tag{4}$$

$$c_{i,2}^{\overline{\gamma}} = g^{\overline{z_i}} h^{\overline{t_i}} \tag{5}$$

Now, eq. (4) and the 2-fROOT assumption implies that $\overline{\gamma} / \gcd(\overline{\gamma}, \overline{t_i}) = 2^{\ell_i'}$ for some $\ell_i' \in \mathbb{N}_0$, as $\overline{\gamma} \neq 0$. From this follows that $\overline{t_i}/\overline{\gamma} = r_i/2^{\ell_i'}$ for some appropriate $r_i \in \mathbb{Z}$. From eq. (5), we find

$$\begin{aligned} & c_{i,2}^{\overline{\gamma}} = g^{\overline{z_i}} h^{\overline{t_i}} \\ \implies & c_{i,2}^{\overline{\gamma}} h^{-\overline{t_i}} = g^{\overline{z_i}} \\ \implies & (c_{i,2} h^{-r_i/2^{\ell_i'}})^{\overline{\gamma}} = g^{\overline{z_i}} \end{aligned}$$

As $u = c_{i,2} h^{-r_i/2^{\ell_i'}}$ is efficiently computable, since square roots can be efficiently computed in our setting, the 2-fROOT assumption yields again that $\overline{z_i}/\overline{\gamma} = y_i/2^{\ell_i}$ for some appropriate $y_i \in \mathbb{Z}$ and $\ell_i \in \mathbb{N}_0$. Without loss of generality, we choose all $\ell_i' = \ell_i \leq \log(C)$ minimal.

We now set $\mu_{i,1} := g^{-\frac{\overline{t_i}}{\overline{\gamma}}} c_{i,1}$ and $\mu_{i,2} := g^{-\frac{\overline{z_i}}{\overline{\gamma}}} h^{-\frac{\overline{t_i}}{\overline{\gamma}}} c_{i,2}$ which are computable since square roots can be computed efficiently in this setting. Because of eqs. (4) and (5), it holds that $\mu_{i,1}$ and $\mu_{i,2}$ vanish after taking it to the power of $\overline{\gamma} \neq 0$ as:

$$\mu_{i,1}^{\overline{\gamma}} = g^{-\overline{t_i}} c_{i,1}^{\overline{\gamma}} = 1,$$
$$\mu_{i,2}^{\overline{\gamma}} = g^{-\overline{z_i}} h^{-\overline{t_i}} c_{i,2}^{\overline{\gamma}} = 1.$$

It follows that $\mu_{i,1} = \mu_{i,2} = 1$ with overwhelming probability by the ORD assumption. Thus, we can set $x_i = \lfloor \frac{y_i}{2^{\ell_i}} \rceil$ and find that c_i opens to message x_i with opening (y_i, ℓ_i, r_i), as verification passes:

$$\begin{aligned} & \mu_{i,1} = 1 \\ \implies & g^{-r_i/2^{\ell_i'}} c_{i,1} = 1 \\ \implies & c_{i,1} = g^{r_i/2^{\ell_i'}} \end{aligned}$$

$$\begin{aligned} & \mu_{i,2} = 1 \\ \implies & g^{-y_i/2^{\ell_i}} h^{-r_i/2^{\ell_i'}} c_{i,2} = 1 \\ \implies & c_{i,2} = g^{y_i/2^{\ell_i}} h^{r_i/2^{\ell_i'}} \end{aligned}$$

Lastly, we argue that x is indeed in the specified range. We obtain similarly as in the proof of theorem 3:

$$\begin{aligned} f = \tilde{f} \implies & h^{\tau} \cdot g^{\gamma} \cdot c_{0,2}^{4(\gamma B - z_0)} \cdot \prod_{i=1..3} c_{i,2}^{-z_i} = h^{\tilde{\tau}} \cdot g^{\tilde{\gamma}} \cdot c_{0,2}^{4(\tilde{\gamma} B - \tilde{z_0})} \cdot \prod_{i=1..3} c_{i,2}^{-\tilde{z_i}} \\ \implies & h^{\overline{\tau}} \cdot g^{\overline{\gamma}} \cdot c_{0,2}^{4(\overline{\gamma} B - \overline{z_0})} \cdot \prod_{i=1..3} c_{i,2}^{-\overline{z_i}} \\ \implies & h^{\overline{\tau}} \cdot g^{\overline{\gamma}} \cdot (g^{y_0/2^{\ell_0}} h^{r_0/2^{\ell_0}})^{4(\overline{\gamma} B - \overline{z_0})} \cdot \prod_{i=1..3} (g^{y_i/2^{\ell_i}} h^{r_i/2^{\ell_i}})^{-\overline{z_i}} \\ \implies & g^{\overline{\gamma} + 4\frac{y_0}{2^{\ell_0}}(\overline{\gamma} B - \overline{z_0}) - \sum_{i=1}^{3} \frac{y_i}{2^{\ell_i}} z_i} = h^{\overline{\tau} + 4\frac{r_0}{2^{\ell_0}} \cdot (\overline{\gamma} B - \overline{z_0}) - \sum_{i=1}^{3} \frac{r_i}{2^{\ell_i}} \overline{z_i}} \end{aligned}$$

After multiplying with $\overline{\gamma}$, we can use lemma 7 because 2^{ℓ_i} divides $\overline{\gamma}$, and we obtain $\overline{\gamma}^2 + \gamma 4\frac{y_0}{2^{\ell_0}}(\overline{\gamma} B - \overline{z_0}) = \sum_{i=1}^{3} \overline{z_i}^2 \geq 0$. After dividing out $\overline{\gamma}^2$ again, we obtain:

$$1 + 4\frac{y_0}{2^{\ell_0}}(B - \frac{y_0}{2^{\ell_0}}) \geq 0$$

By lemma 13, this implies $\lfloor \frac{y_0}{2^{\ell_0}} \rceil \in [0, B]$ as claimed.

Theorem 9. *The scheme* $\mathsf{RP}_{\mathsf{CG}}$ *is honest-verifier zero-knowledge under the hiding property of the commitment scheme* CG*.*

Proof. This follows similarly as in theorem 2 and we ommit the details.

Acknowledgments

We thank Muhammed Esgin for helpful comments, especially concerning the choice of parameters and for pointing out a calculation error in a previous version of this work in the lattice setting. We also thank Abram, Damgård, Orlandi, and Scholl for pointing out the lack of invertible sampling in class groups of imaginary quadratic orders.

References

[Abr+22] Damiano Abram, Ivan Damgård, Claudio Orlandi, and Peter Scholl. "An Algebraic Framework for Silent Preprocessing with Trustless Setup and Active Security". In: *IACR Cryptol. ePrint Arch.* (2022).

[AC20] Thomas Attema and Ronald Cramer. "Compressed Σ-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics". In: *CRYPTO 2020, Part III.* Vol. 12172. LNCS. Springer, Heidelberg, Aug. 2020.

[Bau+18] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. "More Efficient Commitments from Structured Lattice Assumptions". In: *SCN 18*. Vol. 11035. LNCS. Springer, Heidelberg, Sept. 2018.

[Ben] Daniel Benarroch. *Diving into the zk-SNARKs Setup Phase*. `https://medium.com/qed-it/diving-into-the-snarks-setup-phase-b7660242a0d7`.

[Ben+14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. "Better Zero-Knowledge Proofs for Lattice Encryption and Their Application to Group Signatures". In: *ASIACRYPT 2014, Part I*. Vol. 8873. LNCS. Springer, Heidelberg, Dec. 2014.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. "Transparent SNARKs from DARK Compilers". In: *EUROCRYPT 2020, Part I*. Vol. 12105. LNCS. Springer, Heidelberg, May 2020.

[Blö+19] Johannes Blömer, Jan Bobolz, Denis Diemert, and Fabian Eidens. "Updatable Anonymous Credentials and Applications to Incentive Systems". In: *ACM CCS 2019*. ACM Press, Nov. 2019.

[Boo+20] Jonathan Bootle, Anja Lehmann, Vadim Lyubashevsky, and Gregor Seiler. "Compact Privacy Protocols from Post-quantum and Timed Classical Assumptions". In: *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*. Springer, Heidelberg, 2020.

[Bou00] Fabrice Boudot. "Efficient Proofs that a Committed Number Lies in an Interval". In: *EUROCRYPT 2000*. Vol. 1807. LNCS. Springer, Heidelberg, May 2000.

[Bri+88] Ernest F. Brickell, David Chaum, Ivan Damgård, and Jeroen van de Graaf. "Gradual and Verifiable Release of a Secret". In: *CRYPTO'87*. Vol. 293. LNCS. Springer, Heidelberg, Aug. 1988.

[BS96] Wieb Bosma and Peter Stevenhagen. "On the computation of quadratic 2-class groups". en. In: *Journal de Théorie des Nombres de Bordeaux* 8.2 (1996). URL: `http://www.numdam.org/item/JTNB_1996__8_2_283_0/`.

[Bün+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*. IEEE Computer Society Press, May 2018.

[Bün+20] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. "Zether: Towards Privacy in a Smart Contract World". In: *FC 2020*. Vol. 12059. LNCS. Springer, Heidelberg, Feb. 2020.

[CCs08] Jan Camenisch, Rafik Chaabouni, and abhi shelat. "Efficient Protocols for Set Membership and Range Proofs". In: *ASIACRYPT 2008*. Vol. 5350. LNCS. Springer, Heidelberg, Dec. 2008.

[Cec+17] Ethan Cecchetti, Fan Zhang, Yan Ji, Ahmed Kosba, Ari Juels, and Elaine Shi. "Solidus: Confidential distributed ledger transactions via PVORM". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017.

[Cha90] David Chaum. "Showing Credentials without Identification Transferring Signatures between Unconditionally Unlinkable Pseudonyms". In: *AUSCRYPT'90*. Vol. 453. LNCS. Springer, Heidelberg, Jan. 1990.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. "Compact E-Cash". In: *EUROCRYPT 2005*. Vol. 3494. LNCS. Springer, Heidelberg, May 2005.

[Chu+20] Heewon Chung, Kyoohyung Han, Chanyang Ju, Myungsun Kim, and Jae Hong Seo. *Bulletproofs+: Shorter Proofs for Privacy-Enhanced Distributed Ledger*. Cryptology ePrint Archive, Report 2020/735. `https://eprint.iacr.org/2020/735`. 2020.

[CPP17] Geoffroy Couteau, Thomas Peters, and David Pointcheval. "Removing the Strong RSA Assumption from Arguments over the Integers". In: *EUROCRYPT 2017, Part II*. Vol. 10211. LNCS. Springer, Heidelberg, 2017.

[Cro+16] Kyle Croman et al. "On scaling decentralized blockchains". In: *International conference on financial cryptography and data security*. Springer. 2016.

[DF02] Ivan Damgård and Eiichiro Fujisaki. "A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order". In: *ASIACRYPT 2002*. Vol. 2501. LNCS. Springer, Heidelberg, Dec. 2002.

[ElG84] Taher ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". In: *CRYPTO'84*. Vol. 196. LNCS. Springer, Heidelberg, Aug. 1984.

[ENS20] Muhammed F Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. "Practical exact proofs from lattices: New techniques to exploit fully-splitting rings". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2020.

[Esg+19] Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. "Lattice-Based Zero-Knowledge Proofs: New Techniques for Shorter and Faster Constructions and Applications". In: *CRYPTO 2019, Part I*. Vol. 11692. LNCS. Springer, Heidelberg, Aug. 2019.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. "Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations". In: *CRYPTO'97*. Vol. 1294. LNCS. Springer, Heidelberg, Aug. 1997.

[FSW03] Pierre-Alain Fouque, Jacques Stern, and Jan-Geert Wackers. "CryptoComputing with Rationals". In: *FC 2002*. Vol. 2357. LNCS. Springer, Heidelberg, Mar. 2003.

[GI08] Jens Groth and Yuval Ishai. "Sub-linear Zero-Knowledge Argument for Correctness of a Shuffle". In: *EUROCRYPT 2008*. Vol. 4965. LNCS. Springer, Heidelberg, Apr. 2008.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof Systems". In: *SIAM Journal on Computing* 18.1 (1989).

[GN08] Nicolas Gama and Phong Q. Nguyen. "Predicting Lattice Reduction". In: *EUROCRYPT 2008*. Vol. 4965. LNCS. Springer, Heidelberg, Apr. 2008.

[Gro05] Jens Groth. "Non-interactive Zero-Knowledge Arguments for Voting". In: *ACNS 05*. Vol. 3531. LNCS. Springer, Heidelberg, June 2005.

[Gro11] Jens Groth. "Efficient Zero-Knowledge Arguments from Two-Tiered Homomorphic Commitments". In: *ASIACRYPT 2011*. Vol. 7073. LNCS. Springer, Heidelberg, Dec. 2011.

[HKR19] Max Hoffmann, Michael Klooß, and Andy Rupp. "Efficient Zero-Knowledge Arguments in the Discrete Log Setting, Revisited". In: *ACM CCS 2019*. ACM Press, Nov. 2019.

[Hof+17] Jeffrey Hoffstein, Jill Pipher, William Whyte, and Zhenfei Zhang. *A signature scheme from Learning with Truncation*. Cryptology ePrint Archive, Report 2017/995. `https://eprint.iacr.org/2017/995`. 2017.

[Hof+20] Max Hoffmann, Michael Klooß, Markus Raiber, and Andy Rupp. "Black-Box Wallets: Fast Anonymous Two-Way Payments for Constrained Devices". In: *PoPETs* 2020.1 (Jan. 2020).

[KK04] Takeshi Koshiba and Kaoru Kurosawa. "Short Exponent Diffie-Hellman Problems". In: *PKC 2004*. Vol. 2947. LNCS. Springer, Heidelberg, Mar. 2004.

[Lin03] Yehuda Lindell. "Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation". In: *Journal of Cryptology* 16.3 (June 2003).

[Lip03] Helger Lipmaa. "On Diophantine Complexity and Statistical Zero-Knowledge Arguments". In: *ASIACRYPT 2003*. Vol. 2894. LNCS. Springer, Heidelberg, 2003.

[LM19] Russell W. F. Lai and Giulio Malavolta. "Subvector Commitments with Application to Succinct Arguments". In: *CRYPTO 2019, Part I*. Vol. 11692. LNCS. Springer, Heidelberg, Aug. 2019.

[LN17] Vadim Lyubashevsky and Gregory Neven. "One-Shot Verifiable Encryption from Lattices". In: *EUROCRYPT 2017, Part I*. Vol. 10210. LNCS. Springer, Heidelberg, 2017.

[LS18] Vadim Lyubashevsky and Gregor Seiler. "Short, Invertible Elements in Partially Splitting Cyclotomic Rings and Applications to Lattice-Based Zero-Knowledge Proofs". In: *EUROCRYPT 2018, Part I*. Vol. 10820. LNCS. Springer, Heidelberg, 2018.

[Lyu09] Vadim Lyubashevsky. "Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures". In: *ASIACRYPT 2009*. Vol. 5912. LNCS. Springer, Heidelberg, Dec. 2009.

[Lyu12] Vadim Lyubashevsky. "Lattice Signatures without Trapdoors". In: *EUROCRYPT 2012*. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012.

[MIO18] ALESSANDRO MIOLA. "Addressing privacy and fungibility issues in bitcoin: confidential transactions". In: (2018).

[MW17] Daniele Micciancio and Michael Walter. "Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time". In: *CRYPTO 2017, Part II*. Vol. 10402. LNCS. Springer, Heidelberg, Aug. 2017.

[Ped92] Torben P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *CRYPTO'91*. Vol. 576. LNCS. Springer, Heidelberg, Aug. 1992.

[PS00] David Pointcheval and Jacques Stern. "Security Arguments for Digital Signatures and Blind Signatures". In: *Journal of Cryptology* 13.3 (June 2000).

[PS19] Paul Pollack and Peter Schorn. "Dirichlet's proof of the three-square theorem: An algorithmic perspective". In: *Math. Comput.* 88.316 (2019). URL: `https://doi.org/10.1090/mcom/3349`.

[PS96] David Pointcheval and Jacques Stern. "Security Proofs for Signature Schemes". In: *EUROCRYPT'96*. Vol. 1070. LNCS. Springer, Heidelberg, May 1996.

[RS86] Michael O. Rabin and Jeffery O. Shallit. "Randomized Algorithms in Number Theory". In: vol. 39. S1. 1986.

[Sle] Greg Slepak. *How To Compromise Zcash And Take Over The World*. `https://blog.okturtles.org/2016/09/how-to-compromise-zcash-and-take-over-the-world/`.

[Yan+19] Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. "Efficient Lattice-Based Zero-Knowledge Arguments with Standard Soundness: Construction and Applications". In: *CRYPTO 2019, Part I*. Vol. 11692. LNCS. Springer, Heidelberg, Aug. 2019.

Supplementary Material

A Remark on the Ring LWE setting.

We simplify the setting and keep the details minimal, so we might skip some necessary properties of the ring R or other entities. At first, we present the basics for Ring LWE and Ring SIS and for understanding the advantage of the setting and then detail the problem of our approach in the setting.

Let $R = \mathbb{Z}[X]/\langle X^d + 1\rangle$ and $R_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$. The Ring LWE and Ring SIS problems are now defined over the ring R_q. So for a random matrix $\boldsymbol{A} \xleftarrow{\$} \mathbb{R}_q^{(m-n)\times n}$, the Ring LWE problem asks to distinguish between $\vec{b} = \boldsymbol{A} \cdot \vec{s} + \vec{e}$ with short $\vec{s} \in \mathbb{R}_q^n, \vec{e} \in \mathbb{R}_q^{m-n}$ and a random $\vec{b} \xleftarrow{\$} \mathbb{R}_q^{(m-n)}$. Similarly, the Ring SIS problem asks to find a short solution $\vec{z}$ for $\boldsymbol{A} \cdot \vec{z} = \vec{0}$. The commitment scheme of [Bau+18] has the same structure as presented in definition 13, i.e. $\vec{c} = \boldsymbol{A} \cdot \vec{r} + (\vec{0} \parallel \vec{x})$ for some matrix $\boldsymbol{A}$ and short vector $\vec{r}$ with entries in R_q. Also, it has a relaxed opening in the form of checking $f \cdot \vec{c} = \boldsymbol{A} \cdot \vec{r} + f \cdot (\vec{0} \parallel \vec{x})$ for some small $f \in R_q$.

The structure of the zero-knowledge scheme for proving knowledge of the opening of a commitment is similar, since most needed results for the zero-knowledge can directly be translated to the different setting. For example lemma 10 and lemma 11 can just be applied on the coefficients of the polynomials which leads to identical statements. This also applies for the general structure of the range proof, only that the vectors in $\mathbb{Z}_q$ are now vectors of polynomials in R_q. The big difference is that multiplication is defined over the Ring R_q and this allows for some optimizations.

An important optimization is the way the challenge space $\mathcal{C}$ for zero-knowledge schemes can be defined. [LS18] shows that short, non-zero elements in R_q can be inverted. So setting $\mathcal{C} = \{\gamma \in R_q \mid \gamma \text{ short}\}$ and keeping $\#\mathcal{C} \geq 2^\lambda$ allows for proofs with negligible soundness error in one round. Since $\gamma - \gamma' \neq 0$ will be short and thus invertible in R_q for $\gamma, \gamma' \in \mathcal{C}$, the challenge difference can thus be used as (part of the) opening for the commitment scheme.

Further, using the *NTT coefficient* representation of $f \in \mathbb{R}_q$ allows for efficiently proving relations in $\mathbb{Z}_q^d$. We now assume that $q \equiv 1 \mod 2d$. This can be generalized in order to have a slightly different vector representation in $\mathbb{Z}_q$. We refer to [ENS20] for more details on the NTT transformation. We identify f with its NTT representation $\vec{\hat{f}} \in \mathbb{Z}_q^d$, where $\hat{f}_i = f[r_i]$ and r_i is the *i-th* primitive $2d$*-th* root of unity. This representation has the convenient property that a multiplication $f \cdot g$ of two vectors $f, g \in R_q$ corresponds to a component-wise multiplication of $\vec{\hat{f}} \cdot \vec{\hat{g}}$. This also holds for component-wise addition. This allows us to prove the relation $4(x-a)(b-x)+1 = \sum_{i=1}^3 x_i^2$ in R_q and retain the relation component-wise in the respective NTT vectors in $\mathbb{Z}_q^d$. The problem is that despite $\gamma \in \mathcal{C}$ being short, there is no guarantee that $\vec{\hat{\gamma}}$ is also short in $\mathbb{Z}_q^d$. Thus we can not argue in the same way as in the standard lattice setting, where we necessarily use the fact that $\gamma \in [-p, p]$ is short for proving that $x \in [a, b+1]$. So in order to utilize the same range proof technique in the Ring LWE setting, we have to find a challenge space $\mathcal{C}$ such that elements $\gamma \in \mathcal{C}$ are simultaneously short in R_q and their NTT representation $\vec{\hat{c}}$ is short in $\mathbb{Z}_q^d$. Note that $\gamma - \gamma'$ invertible for $\gamma, \gamma' \in \mathcal{C}$ is not sufficient. That is because the binding property of the commitment relies on $\gamma - \gamma'$ being short in the proof of soundness in a zero-knowledge scheme.

If $\mathcal{C}$ has this desired property, the verification of shortness of the masked witnesses $z_i = m + \gamma \cdot x$ can directly be performed for $\vec{\hat{z}_i}$ and we could construct a range proof with the techniques introduced in this work.

B Changelog

From 2021-06-22 to current: We were kindly informed by Abram, Damgård, Orlandi, and Scholl [Abr+22] that there is no (commonly known) algorithm Sample for sampling uniformly random group elements in class groups of imaginary quadratic orders which allows efficient invertible sampling. This was necessary to achieve transparent setup, since it was implicitly exploited in the reductions to set up the (Pedersen) commitment key with known dlog $h = g^x$, and would

then explain h as $h = \mathsf{Sample}(1^\lambda, \mathbb{G}; \rho)$ where ρ is distributed uniformly conditioned on sampling outputting h. Thus, while technically correct without transparent, the reductions were not applicable to the claimed setting of *transparent* setup.

We are now more explicit about the necessity of invertible sampling, especially in the class group setting. To resolve the problem in class groups, we assume the DXDH assumption (with public coins) from the work of Abram et al. [Abr+22] and switching to ElGamal commitments. As an additional benefit, we no longer require $C(k)$-roughness in the class group (which limited challenge space to polynomial for provable security). Thus, we have removed the Pedersen commitment based instantiation altogether (although it can be proven secure by making stronger assumptions). Besides the switch from Pedersen to ElGamal commitments, the range proof is essentially unchanged.

Lastly, we have sharpened lemma 13 slightly. Since this only affects $B = 1$, i.e. binary ranges, this is of little consequence.

From 2021-04-27 to 2021-06-22: We were kindly informed by Muhammed Esgin that there was an error in the proof size calculation in the lattice setting (see appendix C). We updated the calculation and the reported proof sizes accordingly.

C Script For Proof Size Computation

Proof Size in the DLOG Setting In the following, we supply the Python scripts that were used to calculate the proof sizes. First, we present the script used to compute the concrete proof size and determine the parameters of the scheme $\mathsf{RP}_{\mathsf{Log}}$ defined in section 5. The following modifications to $\mathsf{RP}_{\mathsf{Log}}$ were applied:

- Rejection sampling for m_i, s_i, σ,
- Fiat-Shamir transformation for non-interactivity.
- Avoid resending commitments $\vec{c}_i$ in repetitions.

Listing 1.1. The Python script used to determine the parameters and compute proof size of the optimized non-interactive version of $\mathsf{RP}_{\mathsf{Log}}$ established in section 5

```
from math import ceil, sqrt, pi, exp
import math

# colors
HEADER = '\033[95m'
FAIL = '\033[91m'
ENDC = '\033[0m'

def log(x):
    return math.log(x, 2)

def getRejSamplingM(alpha):
    return exp(13.3/alpha + 1/(2*alpha*alpha))

def compute_proof_size(secpar, C, B):
    """
    Computes the size of the proof for given security parameter secpar,
        challenge size C and maximal range B, each in bit.
    """
    # ------------------------------------------------
    # ------------------ Parameters ------------------
    # ------------------------------------------------
    # Constant alpha from the rejection sampling lemma.
    alpha = 256
    M = getRejSamplingM(alpha)
    # rounds necessary for negligible error
    rounds = ceil(secpar/C)
    # uniformly random masking overhead
    L = secpar
```

```
    # rejection sampling overhead
    L_prime = ceil(log(sqrt(2*secpar) * alpha))
    # modulus of the group [bit]
    q = 2*L_prime + 2*C + 2*B + 6
    # hash output size [bit]
    Hash_size = 2*secpar

    # -----------------------------------------------
    # ----------------- Scheme Size -----------------
    # -----------------------------------------------
    comsize = q
    mask_z = L_prime + B + C
    mask_t = L_prime + 3*secpar
    tau = L_prime + B + 3*secpar + 2

    proof_size = 3*comsize + rounds*(Hash_size + 4*(mask_z + mask_t) + tau)
    proof_size_B = ceil(proof_size / 8)
    print("M                          : {:-9}".format(round(M,4)))
    print("L'                         : {:-9}".format(round(L_prime,4)))
    print("Range                [bit]: {:-9}".format(B))
    print("Security Parameter   [bit]: {:-9}".format(secpar))
    print("Challenge Size       [bit]: {:-9}".format(C))
    print("Group Size           [bit]: {:-9}".format(q))
    print("")
    print("Proof Size             [B]: {:-9}".format(proof_size_B))

if __name__ == "__main__":
    secpars = [80,100,128]
    Cs = [80, 100, 128]
    Bs = [32, 64]

    for B in Bs:
        for (secpar,C) in zip(secpars, Cs):
            print("----------------------------------------")
            compute_proof_size(secpar, C, B)
            print("----------------------------------------")
```

Proof Size in the Lattice Setting We provide the Python script used to compute the concrete proof size and parameters of the lattice scheme $\mathsf{RP}_{\mathsf{Lat}}$ of section 6. It checks whether the necessary hardness assumptions are fulfilled by the chosen parameters. The following optimizations from section 6.4 were used:

- Hash function to avoid sending the mask commitments.
- Avoid resending commitments $\vec{c}_i$ in repetitions.
- Rejection sampling to mask $\vec{x}_i$.

Listing 1.2. The Python script used to determine the parameters and compute proof size of the optimized non-interactive version of $\mathsf{RP}_{\mathsf{Lat}}$ established in section 6

```
from math import ceil, sqrt, pi, exp
import math

# colors
HEADER = '\033[95m'
FAIL = '\033[91m'
ENDC = '\033[0m'

def log(x):
    return math.log(x, 2)
```

```
def bit_size(x):
    return ceil(log(x))

def get_rhf_sis(n, m, q, beta):
    a = log(beta)**2
    b = 4*n*q
    temp = a / b
    return 2**temp

# returns the output probability per round
def get_output_P(M_x, M_r):
    return (1/M_x * 1/M_r)

def get_rhf_lwe(n, m, q, sigma):
    a = (log(sigma*sqrt(2*pi))-log(5.31)-q)**2
    b = 4*n*q
    temp = a / b
    return 2**temp

def compute_proof_size(secpar, l1, l2, C, B, req_RHF, alpha_mod, n):
    """
    Computes the size of the proof for given security parameter secpar,
        challenge size C and maximal range B, each in bit.
    """
    # ------------------------------------------------
    # ------------------ Parameters ------------------
    # ------------------------------------------------
    # parameters of the commitment scheme
    l = l1+l2+n
    rounds = ceil(secpar/C)

    n_rej = n
    # for having good output probability, mask all x_i at once for small n
    if (n_rej < 140):
        n_rej = 4*n
    L_x = bit_size(2*(n_rej**(1.5)))
    M_x = exp(13.3/log(n_rej) + 1/(2*log(n_rej)*log(n_rej)))
    M_r = exp(13.3/log(l) + 1/(2*log(l)*log(l)))

    q = L_x + 2*C + 2*B + 6

    # standard deviations
    alpha = ceil(q*alpha_mod)
    sigma = ceil(max(sqrt(2*l2/pi), 2**(q-alpha)/sqrt(2*pi)))
    sigma_rej = ceil(2*(2**C)*sqrt(l)*log(l)*sigma)

    # size of the hash output [bit]
    Hash_size = 2*secpar

    # ------------------------------------------------
    # ----------------- HARDNESS ---------------------
    # ------------------------------------------------

    # Bound of the randomness of the commitment scheme
    T = 4*sqrt(l)*(sigma_rej+2**C+sigma)
    # RHF for the commitment scheme's SIS assumption
    SIS_RHF = get_rhf_sis(l1,l,q,4*(2**C)*T)
    # RHF for the commitment scheme's LWE assumption
    LWE_RHF = get_rhf_lwe(l2,l,q,sigma)

    # RHF needs to be smaller than the required RHF
```

```
    if LWE_RHF >= req_RHF or q > l2 or SIS_RHF >= req_RHF:
        print("Required RHF: " + FAIL + str(req_RHF) + ENDC)
    print("LWE RHF:       " + str(LWE_RHF))
    print("SIS RHF:       " + str(SIS_RHF))

    # -----------------------------------------------
    # ----------------- Scheme Size -----------------
    # -----------------------------------------------
    comsize = (l1+n)*q
    mask_z = n*(B + C + L_x)
    mask_t = l*bit_size(T)

    # compute proof size
    proof_size = 4*comsize + rounds*(Hash_size + 4*mask_z + 5*mask_t)
    proof_size_mB = (proof_size / (8 * 1000 * 1000))

    print("Challenge Size        [bit]: {:-9}".format(C))
    print("Modulus Size          [bit]: {:-9}".format(q))
    print("l_1                        : {:-9}".format(l1))
    print("l_2                        : {:-9}".format(l2))
    print("n                          : {:-9}".format(n))
    print("sigma                 [bit]: {:-9}".format(bit_size(sigma)))
    print("")
    print("Total Proof Size       [mB]: {:-9}".format(proof_size_mB))
    print("Success Probability     [%]: {:-9}".format(get_output_P(M_x, M_r))
        )


if __name__ == "__main__":
    secpars = [80,128]
    ps = [20, 16]
    Bs = [32, 64]
    # note: change l1, l2 and the alpha modifier to fine-tune the difficulty
        of LWE and SIS
    l1s = [[2655, 3180], [3120, 3760]]
    l2s = [[2830, 3620], [3420, 4510]]
    ns = [[1000, 1000], [1000, 1000]]
    alpha_mods = [[0.72, 0.71], [0.65, 0.65]]
    RHFs = [1.0048, 1.0035]

    for i, B in enumerate(Bs):
        for secpar, C, rhf, l1, l2, alpha_mod, n in zip(secpars, ps, RHFs,
            l1s[i], l2s[i], alpha_mods[i], ns[i]):
            print("-----------------------------------------")
            print("Security Parameter    [bit]: {:-9}".format(secpar))
            print("Challenge Size        [bit]: {:-9}".format(C))
            print("Range                 [bit]: {:-9}".format(B))
            print("")
            compute_proof_size(secpar, l1, l2, C, B, rhf, alpha_mod, n)
            print("-----------------------------------------")
```