Succinct Diophantine-Satisfiability Arguments

Patrick Towa[1,2], Damien Vergnaud[3,4]

[1] IBM Research – Zurich
[2] DIENS, École Normale Supérieure, CNRS, PSL University, Paris, France
[3] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[4] Institut Universitaire de France

Abstract. A *Diophantine equation* is a multi-variate polynomial equation with integer coefficients, and it is satisfiable if it has a solution with all unknowns taking integer values. Davis, Putnam, Robinson and Matiyasevich showed that the general Diophantine satisfiability problem is undecidable (giving a negative answer to Hilbert's tenth problem) but it is nevertheless possible to argue in zero-knowledge the knowledge of a solution, if a solution is known to a prover.

We provide the first succinct honest-verifier zero-knowledge argument for the satisfiability of Diophantine equations with a communication complexity and a round complexity that grows logarithmically in the size of the polynomial equation. The security of our argument relies on standard assumptions on hidden-order groups. As the argument requires to commit to integers, we introduce a new integer-commitment scheme that has much smaller parameters than Damgård and Fujisaki's scheme. We finally show how to succinctly argue knowledge of solutions to several NP-complete problems and cryptographic problems by encoding them as Diophantine equations.

Table of Contents

1 Introduction

A *Diophantine equation* is a multi-variate polynomial equation with integer coefficients, and it is satisfiable if it has a solution with all unknowns taking integer values. Davis, Putnam, Robinson and Matiyasevich [29] showed that any computational problem can be modeled as finding a solution of such equations, thereby proving that the general Diophantine satisfiability problem is undecidable and giving a negative answer to Hilbert's tenth problem. For instance, several classical NP-problems such as 3-SAT, Graph 3-colorability or Integer Linear Programming can be readily encoded as Diophantine equations. Several cryptographic problems such as proving knowledge of an RSA signature, that a committed value is non-negative or that encrypted votes are honestly shuffled by a mix-net, can also be encoded as Diophantine equations.

Efficient zero-knowledge arguments of knowledge of solutions to Diophantine equations, if a solution is known to a party, can thus be useful for many practical cryptographic tasks; and doing so requires to do zero-knowledge proofs on committed integers.

1.1 Prior Work

Integer Commitments. Fujisaki and Okamoto [19] presented the first efficient integer commitment scheme and also suggested a zero-knowledge protocol for verifying multiplicative relations over committed values. Such a commitment scheme allows to commit to any $x \in \mathbb{Z}$ in a group of unknown order, with a Pedersen-like commitment scheme. This makes the security analysis more intricate since division modulo the unknown group order cannot be performed in general. As an evidence that this setting is error-prone, it was shown by Michels that the Fujisaki–Okamoto proof system was flawed. Damgård and Fujisaki [15] later proposed a statistically-hiding and computationally binding integer commitment scheme under standard assumptions in a hidden-order group $\mathbb{G}$ with an efficient argument of knowledge of openings to commitments, and arguments of multiplicative relations over committed values. This primitive gives rise to a (honest-verifier) zero-knowledge proof of satisfiability of a Diophantine equation with M multiplication over $\mathbb{Z}$ that requires $\Omega(M)$ integer commitments and requires $\Omega(M)$ proofs of multiplicative relations [15, 28]. These complexities have not been improved since then.

Circuit Satisfiability over $\mathbb{Z}_p$. Similarly, it is possible to design a zero-knowledge proof of satisfiability of an arithmetic circuit over $\mathbb{Z}_p$ using Pedersen's commitment scheme [30] in a group $\mathbb{G}$ of public prime order p. An immediate solution is to use the additive homomorphic properties of Pedersen's commitment and zero-knowledge protocols for proving knowledge of the contents of commitments and for verifying multiplicative relations over committed values [12, 31].

For an arithmetic circuit with M multiplication gates, this protocol requires $\Omega(M)$ commitments and $\Omega(M)$ arguments of multiplication consistency and has a communication complexity of $\Omega(M)$ group elements. In 2009, Groth [23] proposed a sub-linear size zero-knowledge arguments for statements involving linear algebra and used it to reduce this communication complexity to $O\left(\sqrt{M}\right)$ group elements. This breakthrough initiated a decade of progress for zero-knowledge proofs for various statements (see e.g., [5, 8, 10, 24] and references therein). It culminated with the argument system "*Bulletproofs*" proposed by Bünz, Bootle, Boneh, Poelstra, Wuille and Maxwell [10] which permits to prove the satisfiability of such an arithmetic circuit with communication complexity $O(\log(M))$ and round complexity $O(\log(M))$. The corner stone of their protocol is an argument that two committed vectors satisfy an inner-product relation. It has logarithmic communication and round complexity in the vector length; and its security only relies on the discrete-logarithm assumption and does not require a trusted setup.

Circuit satisfiability over any finite field is an NP-complete problem so the "*Bulletproofs*" argument system has a widespread application. However, as mentioned above, in many cryptographic settings, it is desirable to prove statements such as "the committed value x is a valid RSA signature on a message m for an RSA public key (N, e)". In this case, the prover has to convince the verifier that $x^e = H(m) \bmod N$, or in other words that there exists an integer k such that $x^e + kN = H(m)$ where this equality holds over the integers for $|k| \leq N^{e-1}$ and H is some cryptographic hash function. In order to use directly an argument of satisfiability of an arithmetic circuit to prove the knowledge of a pair (x, k) which satisfies this equation, one needs to use a group $\mathbb{G}$ a prime order p with $p > N^e$ (and to additionally prove that $x < N$ and $k < N^e$). For a large e, this approach results in a proof with prohibitive communication complexity.

Moreover, in various settings, such as the Integer-Linear-Programming problem, there is no *a priori* upper-bound on the sizes of the integer solutions during setup when p is defined. Being able to argue on integers instead of residue classes modulo a fixed prime integer then becomes necessary. Besides, generic reductions to circuit satisfiability over prime-order fields for some simple problems naturally defined over the integers may return circuits with a very large number of multiplication gates and even the '*Bulletproofs*" argument system could produce large proofs. Modeling computational problems using Diophantine equations is more versatile, and a succinct argument system for Diophantine satisfiability thus has many potential applications.

1.2 Contributions

We provide the first succinct argument for the satisfiability of Diophantine equations with a communication complexity and a round complexity that grows logarithmically in the size of the polynomial equation[5]. It is statistical honest-verifier zero-knowledge and is extractable under standard computational assumptions over hidden-order groups such as RSA groups or ideal-class groups.

Integer Commitments. Section 3 introduces a new computationally hiding and binding commitment scheme that allows to commit to vectors of integers. It is close to Damgård and Fujisaki's seminal proposal, but has much smaller parameters. Denoting by λ the security parameter and letting $2^{b_{\mathbb{G}}}$ be an upper bound on the group order, the version of our scheme which allows to commit to n integers at once has parameters consisting of $O(b_{\mathbb{G}} + \log n)$ bits instead of $\Omega\left(nb_{\mathbb{G}} \cdot \operatorname{polylog}(\lambda)\right)$ as with the generalized version of Damgård and Fujisaki's scheme.

Damgård and Fujisaki's commitment scheme, for $n = 1$, is a variant of Pedersen's commitment in a hidden-order group $\mathbb{G}$: given two group elements $g, h \in \mathbb{G}$, the commitment to an integer value $x \in \mathbb{Z}$ is $C = g^x h^r$, where r is an integer of appropriate size. The hiding property of their scheme crucially relies on the fact that $g \in \langle h \rangle$, which is not always guaranteed as the group may not be cyclic. Damgård and Fujisaki's proposed a Schnorr-type [31] protocol to prove such statements, but their challenge set is restricted to $\{0, 1\}$ to guarantee soundness under the assumptions on the group. Their protocol must then be repeated logarithmically many times to achieve negligible soundness, and the resulting parameters are large. The situation is worse when n is large as commitments are computed as $g_1^{x_1} \cdots g_n^{x_n} h^r$ and a proof for each g_i must be computed.

Our scheme is based on the observation that proving that $g^2 \in \left\langle h^2 \right\rangle$ can be done more efficiently in a single protocol run under the assumptions on the group. Our commitments are thus computed as $(g^x h^r)^2 \in \mathbb{G}$. We further such how to aggregate the proofs of several such statements to reduce the size of our parameters when n is large.

[5] Our goals and techniques differs completely from those proposed by Bünz, Fisch and Szepieniec [11] where they used what they called *Diophantine Argument of Knowledge (DARK)* to construct a commitment scheme for polynomials over prime finite fields (using the so-called *Kronecker substitution* for determining the coefficients of a polynomial by evaluating it at a single value, see e.g., [20, p. 245]).

Succinct Inner-Product Arguments on Integers. Section 4 presents a succinct argument that two integer vectors committed with our scheme satisfy an inner-product relation. That is, an argument of knowledge of vectors $\mathbf{a}$ and $\mathbf{b} \in \mathbb{Z}^n$ (and of a randomness $r \in \mathbb{Z}$) that open a commitment C and such that $\langle \mathbf{a}, \mathbf{b} \rangle = z$ given a public integer z. Succinct here means that the communication complexity of the prover is of order $O(\ell + \log(n) b_{\mathbb{G}})$, where ℓ is the bit length of the largest witness. The complexity is measured in bits as during the protocol, the prover sends logarithmically many group elements and three integers, but these latter could be arbitrarily large.

The argument of Bünz et al. [10] for inner-product relations over $\mathbb{Z}_p$ is not applicable to integers as their proof of extractability relies on the generalized discrete-logarithm assumption for which there is no equivalent in hidden-order groups that may not even be cyclic, and on the invertibility of elements in $\mathbb{Z}_p^*$ since it requires to solve linear systems over $\mathbb{Z}_p$. Besides, their argument is not zero-knowledge and is on vectors committed with the non-hiding version of Pedersen's scheme (i.e., with nil randomness). Therefore, whenever it is used as a sub-protocol in another one, techniques specific to the larger protocol must always be used to guarantee that it is zero-knowledge. del Pino, Seiler and Lyubashevsky [16] later solved this issue by adapting the argument of Bünz et al. in prime-order groups to make it perfectly honest-verifier zero-knowledge with the full-fledged Pedersen's scheme.

Our protocol uses halve-then-recurse techniques similar to those of Bünz et al. for the Section-3.2 commitment scheme in hidden-order groups and thus allows to succinctly argue on integers, but only uses the integrality of $\mathbb{Z}$ as a ring since one cannot invert modulo the unknown order. (Note that these techniques are themselves inspired by the recursive inner-product argument of Bootle et al. [8].) In particular, we prove that even though one cannot a priori solve in $\mathbb{Z}$ the linear system of Bünz et al. required to prove the extractability of their protocol, one can instead solve a "relaxed" system in $\mathbb{Z}$. Then, under the assumptions on the hidden-order group, we show that the solution to the relaxed system is enough to extract a representation of the commitment in the public bases. In groups with public prime orders, the assumption that discrete-logarithm relations are hard to compute allows to conclude that this representation of the commitment actually leads to a valid witness, but this assumption is not a priori translatable to hidden-order groups. Instead, we prove that a similar assumption in the subgroup generated by a randomly sampled element is weaker than the assumptions on the group, and that suffices to prove the extractability of the protocol. The details on these technical challenges are outlined in Section 4.1.

Furthermore, as the group order is unknown to all parties, the argument is only statistically honest-verifier zero-knowledge. To ensure this property, the randomness range of the prover is carefully[6] adapted to allow for simulatability without knowledge of a witness.

[6] As another evidence that cryptography in hidden-order groups is error prone, Fouque and Poupard [18] broke the RDSA signature from [7] for which this randomness range was not wisely selected.

Succinct Arguments for Multi-Integer Commitments. Section 5 then gives several succinct protocols related to multi-integer commitments. These protocols are important building blocks in our Diophantine satisfiability argument system but may also find applications in other settings.

We show how to succinctly argue knowledge of an opening $(a_1, \ldots, a_n, r)$ to a commitment. The protocol has bit communication complexity $O\left(\ell + \log(n) b_{\mathbb{G}}\right)$. This argument system is based on the same halve-then-recurse techniques as in Section 4. We also propose a protocol that allows to aggregate arguments of knowledge of openings to m such commitments for any $m \geq 2$. With the same notation, the bit communication complexity of this aggregated protocol is only $O\left(\ell + \log(n) b_{\mathbb{G}} + \log(m)\right)$ (i.e., the number of group elements does not increase with m). It is worth mentioning that the techniques used in it can be applied to additively-homomorphic commitment scheme in public-order groups.

We also show how to obtain short parameters for our new integer-vector commitment scheme and the inner-product argument with communication complexity $O\left(\ell + \log(n) b_{\mathbb{G}} + \log(m)\right)$ bits, i.e., arguments that group elements $g_1, \ldots, g_m, h$ for $m \geq 2$ satisfy $g_i^2 \in \left\langle h^2 \right\rangle$ for $i \in \{1, \ldots, m\}$ with communication complexity $O(b_{\mathbb{G}} + \log m)$ bits instead of $O(m b_{\mathbb{G}})$ bits obtained by repeating m times the protocol for one group element. Finally, we show how to succinctly argue knowledge of the same vector of integers in $\mathbb{Z}^n$ committed with our schemes using m different bases for any $m \geq 2$ (and different randomness).

Succinct Arguments for Diophantine Equations. Section 6 presents our succinct protocol to argue satisfiability of Diophantine equations. Our approach is inspired by Skolem's method [32] which consists in reducing the degree of the polynomial by introducing new variables to obtain a new polynomial of degree at most 4, in such a way that the satisfiability of one polynomial implies that of the other. Tailoring Skolem's method to the problem of arguing satisfiability, we show how to reduce the satisfiability of any polynomial in $\mathbb{Z}[x_1, \ldots, x_\nu]$ of total degree δ with μ monomials to the existence of vectors $\mathbf{a}_L = \left[a_{L,1} \cdots a_{L,n}\right]$, $\mathbf{a}_R = \left[a_{R,1} \cdots a_{R,n}\right]$ and $\mathbf{a}_O = \left[a_{O,1} \cdots a_{O,n}\right]$ in $\mathbb{Z}^n$, for $n \leq \nu \lfloor \log \delta \rfloor + (\delta - 1)\mu$, such that $a_{O,i} = a_{L,i} a_{R,i}$ for all $i \in \{1, \ldots, n\}$, and that satisfy $1 \leq Q \leq 1 + 2\nu(\lfloor \log \delta \rfloor - 1) + (\delta - 2)\mu$ linear constraints of the form

$$\langle \mathbf{w}_{L,q}, \mathbf{a}_L \rangle + \langle \mathbf{w}_{R,q}, \mathbf{a}_R \rangle + \langle \mathbf{w}_{O,q}, \mathbf{a}_O \rangle = c_q,$$

where $\mathbf{w}_{L,q}, \mathbf{w}_{R,q}, \mathbf{w}_{O,q} \in \mathbb{Z}^n$ and $c_q \in \mathbb{Z}$ for all $q \in \{1, \ldots, Q\}$. Our reduction is constructive as it allows to infer the vectors and the constraints directly from the original polynomial.

Bootle et al. [8] then Bünz et al. [10] gave an argument system for proving knowledge of vectors in $\mathbb{Z}_p$ (instead of $\mathbb{Z}$) that satisfy such constraints. They use this protocol to argue for the satisfiability of arithmetic circuits over $\mathbb{Z}_p$. Our argument shares similarities with theirs, but again there are key technical differences that arise from the fact that $\mathbb{Z}$ is not a field. Indeed, as one cannot invert nor reduce integers modulo the unknown orders of the bases, we use different techniques notably to prevent the integers involved in the argument from increasing too much, and to ensure consistency between the variables in the entry-wise product and those in the linear constraints. Guaranteeing this latter consistency requires to construct new polynomials for the argument that do not involve inverting integers. Besides, one cannot use their commitment-key switching technique which consists in interpreting g^a as a commitment to xa to the base $g^{x^{-1}}$ in groups of public prime order. Finally, extra precaution must be taken to guarantee the zero-knowledge property as integers are not reduced module p and may carry information about the witness. These challenges and the ways we overcome them are described in details in Section 6.2.

As a result, the communication complexity of our Diophantine-satisfiability argument has a communication-complexity of $O\left(\delta\ell + \min(\nu,\delta)\log\left(\nu+\delta\right)b_{\mathbb{G}} + H\right)$ bits, if the absolute value of all the polynomial coefficients is upper-bounded by 2^H for some integer H. In contrast, the overall communication complexity using Damgård and Fujisaki's multiplication argument is upper-bounded by $O\left(\binom{\nu+\delta}{\delta}\left(\delta\ell + \log\left(\binom{\nu+\delta}{\delta}\right)H + b_{\mathbb{G}}\right)\right)$ and lower-bounded by $\Omega\left(\binom{\nu+\delta}{\delta}(\ell + b_{\mathbb{G}})\right)$.

Applications. Section 7 finally presents several applications of our Diophantine-satisfiability argument. We provide explicit reductions to Diophantine satisfiability for the following problems:

- argument of knowledge of a (possibly committed) RSA e-th root in $\mathbb{Z}_N$ of some public value with in $O\left(\log(\log(e))b_{\mathbb{G}}\right)$ bits. This has application to credential systems when combined with proofs of non-algebraic statements [13];
- argument of knowledge of $O\left(\log(\log p)b_{\mathbb{G}}\right)$ bits for ECDSA signatures with a prime p, and of $O\left(\log(\log q)b_{\mathbb{G}} + \log(\log p)\right)$ bits for DSA signatures with primes p and q. The signed message is public, but can be committed if the argument is combined with proofs of non-algebraic statements [13];
- argument that two committed lists of integers of length n are permutations of each other with $O\left(\ell + \log(n)b_{\mathbb{G}}\right)$ bits;
- argument of satisfiability of a 3-SAT Boolean formula with m clauses and n variables with $O\left(\log(n+m)b_{\mathbb{G}}\right)$ bits;
- argument of satisfiability of Integer-Linear-Programming problem of the form $\mathbf{x} \in \mathbb{N}^n$ and $\mathbf{A}\mathbf{x}^{\mathrm{T}} \geq \mathbf{b}^{\mathrm{T}}$, for $\mathbf{A} \in \mathbb{Z}^{m\times n}$ and $\mathbf{b} \in \mathbb{Z}^m$, with $O\left(\ell + \log(4n+3m)\right.$ $\left.b_{\mathbb{G}} + \log\|\mathbf{A}\|_\infty + \log\|\mathbf{b}\|_\infty\right)$ bits.

2 Preliminaries

This section introduces the notation used throughout the paper, recalls standard assumptions on generators of hidden-order groups, and defines commitment schemes and argument systems.

2.1 Notation

For $x \in \mathbb{Z}$, $|x|$ denotes its absolute value. All logarithms are in base 2. For any two integers $a \leq b \in \mathbb{Z}$, $[\![a;b]\!]$ denotes the set $\{a\}$ if $a = b$ and $\{a, a+1, \ldots, b\}$ if

$a < b$. For an integer $n \geq 1$, $[\![n]\!]$ stands for the set $[\![1;n]\!]$. Given a vector $\mathbf{a} \in \mathbb{Z}^n$, $\mathbf{a}X$ denotes the vector $\left[a_1X\ a_2X\ \cdots\ a_nX\right] \in \mathbb{Z}^n[X]$.

For a given group $(\mathbb{G}, \cdot)$, $T_{\mathbb{G}}$ denotes the binary complexity of computing group operations. For $h \in \mathbb{G}$, $\sqrt{\langle h^2 \rangle}$ denotes the subgroup $\left\{g \in \mathbb{G} \colon \exists \alpha \in \mathbb{Z}, g^2 = h^{2\alpha}\right\}$.

For $\mathbf{g} \in \mathbb{G}^n$, if n is even, set $\mathbf{g}_1 := \left[g_1 \cdots g_{n/2}\right]$ and $\mathbf{g}_2 := \left[g_{n/2+1} \cdots g_n\right]$, and if n is odd, set $\mathbf{g}_1 := \left[g_1 \cdots g_{\lfloor n/2 \rfloor}\ 1_{\mathbb{G}}\right]$ and $\mathbf{g}_2 := \left[g_{\lceil n/2 \rceil} \cdots g_n\right]$. For $\mathbf{a} \in \mathbb{Z}^n$, if n is even, set $\mathbf{a}_1 := \left[a_1 \cdots a_{n/2}\right]$ and $\mathbf{a}_2 := \left[a_{n/2+1} \cdots a_n\right]$, and if n is odd, set $\mathbf{a}_1 := \left[a_1 \cdots a_{\lfloor n/2 \rfloor}\ 0\right]$ and $\mathbf{a}_2 := \left[a_{\lceil n/2 \rceil} \cdots a_n\right]$.

For $n \in \mathbb{N}^*$, $z \in \mathbb{Z}$ and $\mathbf{g} = \left[g_1 \ldots g_n\right] \in \mathbb{G}^n$, let $\mathbf{g}^z := \left[g_1^z \cdots g_n^z\right] \in \mathbb{G}^n$. For $\mathbf{a} = \left[a_1 \ldots a_n\right] \in \mathbb{Z}^n$, define $\mathbf{g}^{\mathbf{a}} := \prod_{i=1}^n g_i^{a_i}$. For $\mathbf{g}$ and $\mathbf{h}$ in $\mathbb{G}^n$, $\mathbf{g} \circ \mathbf{h} \in \mathbb{G}^n$ denotes their Hadamard product, i.e., their component-wise product.

2.2 Hidden-Order-Group Generators and Hardness Assumptions

A hidden-order-group generator G is an algorithm which takes as an input a security parameter 1^λ and returns the description of a finite Abelian group $(\mathbb{G}, \cdot)$ and an integer $P \geq 2$. Integer P is assumed to be smaller than the order of $\mathbb{G}$, but to still be a super-polynomial function of the security parameter. The role of P is mainly to adjust the soundness of the protocols herein, as their challenge spaces will typically be $\left[\!\left[0; P^{\Omega(1)} - 1\right]\!\right]$.

It is also assumed that given the description of $\mathbb{G}$, the group law and the inversion of group elements can be efficiently computed, that group elements can be sampled uniformly at random and that an upper bound $2^{b_{\mathbb{G}}}$ on $\mathrm{ord}(\mathbb{G})$ can be efficiently computed, with $b_{\mathbb{G}} := b_{\mathbb{G}}(\lambda)$ polynomial in λ (it is further assumed that $b_{\mathbb{G}} = \Omega(\lambda)$). Recall that the bit complexity of an elementary operation in a group $\mathbb{G}$ is denoted $T_{\mathbb{G}}$.

The following assumptions are classical for hidden-order-group generators and were introduced by Damgård and Fujisaki [15]. They are best illustrated for P such that natural integers less than P are factorizable in polynomial time in λ (e.g., $\lambda^{\log^{\Omega(1)}(\lambda)}$ given current knowledge in computational number theory), and for $\mathbb{G}$ as the group $\mathbb{Z}_N^*$ for an RSA modulus N with prime factors p and q such that $p = q = 3 \mod 4$, $\gcd(p-1, q-1) = 2$ and the number of divisors of $p - 1$ and $q - 1$ with prime factors less than P is of magnitude $O(\lambda)$. However, these assumptions are believed to also hold over generators of ideal-class groups.

Definition 2.1 (Strong-Root Assumption). *A group generator G satisfies the (T, ε)-strong-root assumption if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs in time at most $T(\lambda)$,*

$$\Pr\left[g^n = h \wedge n > 1 \colon \begin{array}{c} (\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right) \\ h \leftarrow_\$ \mathbb{G} \\ (g, n) \leftarrow \mathcal{A}(\mathbb{G}, P, h) \end{array}\right] \leq \varepsilon(\lambda).$$

This assumption is simply a generalization of the strong RSA assumption [4, 19] to hidden-order groups.

Definition 2.2 (Small-Order Assumption). *A group generator* G *satisfies the* (T,ε)*-small-order assumption if for all* $\lambda \in \mathbb{N}$*, for every adversary* $\mathcal{A}$ *that runs in time at most* $T(\lambda)$*,*

$$\Pr\left[\begin{matrix} g^n = 1_{\mathbb{G}} \wedge g^2 \neq 1 \\ 0 < n < P \end{matrix} : \begin{matrix} (\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right) \\ (g, n) \leftarrow \mathcal{A}(\mathbb{G}, P) \end{matrix}\right] \leq \varepsilon(\lambda).$$

The small-order assumption simply states that it should be hard to find low-order elements in the group (different from $1_{\mathbb{G}}$), except for square roots of unity which may be easy to compute (e.g., -1 in RSA groups). In the group $\mathbb{Z}_N^*$ for $N = pq$ with p and q prime such that $\gcd(p-1, q-1) = 2$, Damgård and Fujisaki [15] showed that factoring N can be reduced to this problem in polynomial time if integers less than P are factorizable in polynomial time in λ.

Definition 2.3 (Orders with Low Dyadic Valuation). *A group generator* G *satisfies the low-dyadic-valuation assumption on orders if for all* $\lambda \in \mathbb{N}$*, for every* $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right)$*, for every* $g \in \mathbb{G}$*,* $\operatorname{ord}(g)$ *is divisible by* 2 *at most once.*

Notice that in the group $\mathbb{Z}_N^*$ for $N = pq$ with p and q prime such that $p = q = 3 \mod 4$, the order of any element is divisible by 2 at most once since 2 divides $p-1$ and $q-1$ exactly once.

Definition 2.4 (Many Rough-Order Elements or μ-Assumption). *An integer is said to be* P*-rough if all its prime factors are greater than or equal to* P*. A group generator* G *satisfies the* μ*-assumption that there are many rough-order elements in the groups generated by* G *(or simply the* μ*-assumption) if for all* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[\operatorname{ord}(h) \textit{ is } P\textit{-rough} : \begin{matrix} (\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right) \\ h \leftarrow_{\$} \mathbb{G} \end{matrix}\right] \geq \mu(\lambda).$$

2.3 Non-interactive Commitments

This section defines commitment schemes. The following definitions are given in a model in which the scheme algorithms (and the adversary) are given access to a random oracle. The reason is that the algorithms may have to check non-interactive proofs computed with a random oracle before carrying on with their computation. Therefore, the number of random-oracle queries made by an adversary can affect the security of the scheme.

Formally, a (non-interactive) commitment scheme consists of the following algorithms.

$\mathsf{Setup}\left(1^\lambda\right) \rightarrow pp$: generates public parameters on the input of a security parameter 1^λ. These parameters are implicit inputs to the other algorithms.

$\mathsf{KG}\left(pp\right) \rightarrow ck$: computes a commitment key on the input of public parameters. The parameters and the commitment key further define a message space denoted $\mathcal{X}_{pp,ck}$.

$\mathsf{Com}\,(ck, x) \to (C, d)$: computes a commitment C to a value x and an opening or decommitment information d on the input of a commitment key ck. It is further assumed that if $x \notin \mathcal{X}_{pp,ck}$, then the algorithm returns $\perp$.

$\mathsf{ComVf}(ck, C, x, d) \to b \in \{0, 1\}$**:** deterministically returns a bit indicating whether the decommitment d is valid (bit 1) for C and x w.r.t. key ck, or not (bit 0). It is assumed that if $C = \perp$ or if $x \notin \mathcal{X}_{pp,ck}$, then it returns 0.

A commitment scheme is *correct* if for all $\lambda \in \mathbb{N}$, for all $pp \leftarrow \mathsf{Setup}\left(1^\lambda\right)$, for all $ck \leftarrow \mathsf{KG}\,(pp)$ and all $x \in \mathcal{X}_{pp,ck}$,

$$\Pr\left[\mathsf{ComVf}(ck, C, x, d) = 1 \colon (C, d) \leftarrow \mathsf{Com}(ck, x)\right] = 1.$$

Give a random oracle $\mathcal{H}$, a commitment scheme is $(T, q_{\mathcal{H}}, \varepsilon)$-*hiding* ($(q_{\mathcal{H}}, \varepsilon)$-statistically hiding) if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs it time at most $T(\lambda)$ (computationally unbounded, i.e., $T(\lambda) = \infty$) and makes at most $q_{\mathcal{H}}$ queries to $\mathcal{H}$,

$$\left|\Pr\left[b = b' \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right) \\ (ck, x_0, x_1, st) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(pp) \\ b \leftarrow_\$ \{0,1\} \\ (C, d) \leftarrow \mathsf{Com}(ck, x_b) \\ b' \leftarrow \mathcal{A}(st, C) \\ \text{for } e \in \{0,1\} \\ \quad\quad \text{if } x_e \in \mathcal{X}_{pp,ck} \text{ and } x_{1-e} \notin \mathcal{X}_{pp,ck} \\ \quad\quad\quad\quad b' \leftarrow_\$ \{0,1\} \end{array}\right] - 1/2\right| \le \varepsilon(\lambda).$$

Note that in the definition, the adversary is the party computing the commitment key. One could consider a weaker variant of the definition with *trusted* key generation in which the key is necessarily honestly generated.

Give a random oracle $\mathcal{H}$, a commitment scheme is $(T, q_{\mathcal{H}}, \varepsilon)$-*binding* if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs in time at most $T(\lambda)$ and makes at most $q_{\mathcal{H}}$ queries to $\mathcal{H}$,

$$\Pr\left[\begin{array}{c} \mathsf{ComVf}(ck, C, x_i, d_i) = 1 \\ \wedge\ x_0 \neq x_1 \end{array} \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right) \\ ck \leftarrow \mathsf{KG}\,(pp) \\ (C, (x_i, d_i)_{i=0,1}) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(pp, ck) \end{array}\right] \le \varepsilon(\lambda).$$

Discussion. The syntax above separates the commitment-key generation algorithm from the setup algorithm, although these are often tacitly combined, especially for commitments in public-order groups. The main reason is that doing so allows to define the hiding property for schemes even when the keys are possibly *invalid.* This question does not arise for schemes with keys that are elements of a prime-order group $\mathbb{G} = \langle g \rangle$ (e.g., Pedersen's scheme [30]) since any element $h \in \mathbb{G}^*$ is a valid commitment key. However, when the scheme is defined over an unknown-order group $\mathbb{G}$ which may not be cyclic, and that keys are elements of the *subgroup* generated by an element (as it is the case for Damgård–Fujisaki commitments recalled in Section 3.1), say h, there may not be an efficient way to test whether another element $g \in \mathbb{G}$ is in $\langle h \rangle$. Computing a commitment with an invalid key may then not guarantee that the commitment is hiding. That is why the definition of the hiding property allows the key to be adversarially generated so that if the definition is satisfied, commitments computed with a potentially invalid key do not reveal information about the committed values.

On the other hand, the definition of the binding property does not consider adversarially generated keys. To understand why, it might be helpful to rather think of an interactive commitment protocol between Alice and Bob. Bob generates a key and sends it to Alice, and Alice commits to a value that she later opens to Bob. It now becomes clear that Alice's committed value should remain hidden before she opens the commitment, and so even if she does not trust Bob's key. Yet, Bob, who needs to ensure that Alice does not later open to a value different from the committed one, is the party who computed the key and thus need not verify that it is valid. The situation is the same with Pedersen's scheme, as its binding property relies on the fact that the discrete-logarithm relation between g and h is unknown to the party who computes the commitment.

2.4 Argument Systems

This section defines argument systems for families of languages. The languages are parametrized by public parameters and Common-Reference String (CRS). As a simple example, given an Abelian group $\mathbb{G}$ (which could be non-cyclic) and an element $h \in \mathbb{G}$ (the parameters) and another element $g \in \langle h \rangle$ (the CRS), consider the language of group elements $C \in \mathbb{G}$ such that there exists $x, y \in \mathbb{Z}$ for which $C = g^x h^y$. This language is clearly parametrized by the parameters and the CRS, and one can give an argument system for this parametrized language in the same vein as what is subsequently done in the paper. However, to lighten the notation, arguments will be (abusively) referred to as arguments for languages rather than arguments for families of languages.

Formally, an argument system (or protocol) for a language $\mathcal{L} = \mathcal{L}_{pp,crs}$ (or equivalently, for the corresponding relation $\mathcal{R} = \mathcal{R}_{pp,crs}$) consists of a quadruple $\Pi = (\mathsf{Setup}, \mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Vf})$ such that $\mathsf{Setup}\left(1^\lambda\right) \rightarrow pp$ returns public parameters on the input of a security parameter, $\mathsf{CRSGen}(pp) \rightarrow crs$ returns a CRS, and $\langle \mathsf{Prove}(crs, x, w) \rightleftharpoons \mathsf{Vf}(crs, x) \rangle \rightarrow (\tau, b) \in \{0,1\}^* \times \{0,1\}$ are interactive algorithms (τ denotes the transcript of the interaction and b the decision bit of Vf). The public parameters are assumed to be tacit inputs to algorithms Prove and Vf, even though they may at times be made explicit for instantiated protocols, especially when the CRS is the empty string (in which case the CRS is omitted from the syntax).

Completeness. Π is complete if for all $\lambda \in \mathbb{N}$, for all $pp \leftarrow \mathsf{Setup}\left(1^\lambda\right)$, $crs \leftarrow \mathsf{CRSGen}\left(pp\right)$, for all $(x, w) \in \mathcal{R}$, $\Pr\left[(*, 1) \leftarrow \langle \mathsf{Prove}(crs, x, w) \rightleftharpoons \mathsf{Vf}(crs, x) \rangle\right] = 1$.

Soundness. Π satisfies (T, ε)-soundness if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs in time at most $T(\lambda)$,

$$\Pr\left[b = 1 \wedge x \notin \mathcal{L} \colon \begin{array}{r} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right); crs \leftarrow \mathsf{CRSGen}\left(pp\right) \\ (st, x) \leftarrow \mathcal{A}(crs) \hfill \\ (\tau, b) \leftarrow \langle \mathcal{A}(st, x) \rightleftharpoons \mathsf{Vf}(crs, x) \rangle \hfill \end{array} \right] \leq \varepsilon(\lambda).$$

This definition of soundness formalizes the idea that an adversary should not be able to convince the verifier that a word x is in $\mathcal{L}$ although it is actually in $\bar{\mathcal{L}} := \{0, 1\}^* \setminus \mathcal{L}$. Groth, Ostrovsky and Sahai [22, 25] relaxed the notion of soundness so that a protocol which satisfies the relaxed notion only guarantees that an adversary cannot convince the verifier that a word is in $\mathcal{L}$ when it is actually in the complement $\bar{\Lambda}$ of language $\Lambda \supseteq \mathcal{L}$ (to be completely formal, Λ should also parametrized by pp and crs). They called this new notion co-soundness or *culpable soundness.* It means that a malicious could still convince the verifier that a word is $\mathcal{L}$ when it is actually in $\Lambda \setminus \mathcal{L}$. However, for many applications, this notion is sufficient.

Formally, a protocol Π for a Language $\mathcal{L}$ satisfies $(T, \varepsilon, \Lambda \supseteq \mathcal{L})$-soundness if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs in time at most $T(\lambda)$,

$$\Pr\left[b = 1 \wedge x \notin \Lambda \colon \begin{array}{r} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right); crs \leftarrow \mathsf{CRSGen}\left(pp\right) \\ (st, x) \leftarrow \mathcal{A}(crs) \hfill \\ (\tau, b) \leftarrow \langle \mathcal{A}(st, x) \rightleftharpoons \mathsf{Vf}(crs, x) \rangle \hfill \end{array} \right] \leq \varepsilon(\lambda).$$

Extractability. Π is $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, \varepsilon)$-extractable if for every deterministic algorithm Prove^* running in time at most $T_{\mathsf{Prove}^*}(\lambda)$, there exists an algorithm (called extractor) $\mathcal{E}(pp, x) \to w$ such that for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ running in time at most $T_{\mathcal{A}}(\lambda)$,

$$* \ \Pr\left[(x, w) \notin \mathcal{R} : \begin{array}{r} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right); crs \leftarrow \mathsf{CRSGen}\left(pp\right) \\ (st, x, s) \leftarrow \mathcal{A}(crs) \hfill \\ w \leftarrow \mathcal{E}^{\langle \mathsf{Prove}^*(crs, x, s) \rightleftharpoons \mathsf{Vf}(crs, x) \rangle}(crs, x) \hfill \end{array} \right] \leq \varepsilon(\lambda),$$

$*$ the running time of $\mathcal{E}$ is at most $T_{\mathcal{E}}$ in expectation, with $T_{\mathcal{E}}$ depending on T_{Prove^*} and

$$\varepsilon_{\mathcal{A}, \mathsf{Prove}^*}(\lambda) := \Pr\left[\beta = 1 : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right); crs \leftarrow \mathsf{CRSGen}\left(pp\right) \\ (st, x, s) \leftarrow \mathcal{A}(crs) \\ (\tau, \beta) \leftarrow \langle \mathsf{Prove}^*(crs, x, s) \rightleftharpoons \mathsf{Vf}(crs, x) \rangle \end{array} \right].$$

The above definition means that if Π is extractable, $\mathcal{E}$ can extract a valid witness from any prover Prove^* with the running time of $\mathcal{E}$ depending on the running time of Prove^* and on the success probability of $(\mathcal{A}, \mathsf{Prove}^*)$, except with probability at most $\varepsilon(\lambda)$. Π is thus an argument of knowledge, which implies that it is sound. The string s given to Prove^* can considered as an internal state which includes its random string.

Similarly to the notion of soundness, the notion of extractability can be extended to a notion of *culpable extractability* w.r.t. a relation $\Sigma \supseteq \mathcal{R}$.

Honest-Verifier Zero-Knowledge. Π is $(T, T_{\mathsf{Sim}}, \varepsilon)$-honest-verifier zero-knowledge ($(T_{\mathsf{Sim}}, \varepsilon)$-statistically honest-verifier zero-knowledge) if there exists an algorithm Sim running in time at most $T_{\mathsf{Sim}}(\lambda)$ such that for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ running in time at most $T(\lambda)$ (for every computationally unbounded adversary $\mathcal{A}$),

$$\left|\Pr\left[(x, w) \in \mathcal{R} \land b = 1 \ : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right) \\ (crs, st, x, w) \leftarrow \mathcal{A}(pp) \\ (\tau, \beta) \leftarrow \langle \mathsf{Prove}(crs, x, w) \rightleftharpoons \mathsf{Vf}(crs, x)\rangle \\ b \leftarrow \mathcal{A}(st, (\tau, \beta)) \end{array}\right] \right.$$
$$\left. - \Pr\left[(x, w) \in \mathcal{R} \land b = 1 : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right) \\ (crs, st, x, w) \leftarrow \mathcal{A}(pp) \\ (\tau, \beta) \leftarrow \mathsf{Sim}(crs, x) \\ b \leftarrow \mathcal{A}(st, (\tau, \beta)) \end{array}\right]\right| \leq \varepsilon(\lambda).$$

Note that the common-reference string is generated by the adversary in this definition. The reasons are the same as for the commitment key in the definition of the hiding property of commitment schemes. A weaker definition in which the common-reference string is honestly generated could also be considered.

Π is said to be *public coin* if all messages sent by Vf are chosen uniformly at random and independently of the messages sent by algorithm Prove.

Interactive Arguments in the Random–Oracle Model. Interactive arguments could also be defined in a model in which the protocol algorithms are given access to a random oracle (in this paper, it will primarily be in case CRSGen needs to compute a non-interactive proof that the CRS is well-formed). The soundness and zero-knowledge properties of the protocol may then be affected by the number of random-oracle queries made by the parties, e.g., if the protocol involves as sub-protocol the Fiat–Shamir (see the next section) non-interactive variant of another protocol, as it is the case in Section 4. The definitions of these properties should then be adapted to include an upper-bound $q_{\mathcal{H}}$ on the number of queries that the adversary can make.

For instance, given a random-oracle $\mathcal{H}$, one can define $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, q_{\mathcal{H}}, \varepsilon)$-*extractability* in the same way as $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, \varepsilon)$-extractability in the standard model was define, but with the difference that $\mathcal{A}$ and Prove^* are given oracle access to $\mathcal{H}$ and can together make at most $q_{\mathcal{H}}(\lambda)$ queries. The other definitions are adapted in a similar fashion.

Fiat–Shamir Heuristic. The Fiat–Shamir heuristic [17] can be used to turn a public-coin, interactive argument system into a non-interactive one in the random-oracle model [6]. Given a random oracle $\mathcal{H}$, the messages of the verifier are computed by evaluating $\mathcal{H}$ at the word and the transcript of the interactive protocol until that point of the computation of the prover. With oracle access to $\mathcal{H}$, the prover can then compute a full transcript (or argument), further denoted π instead of τ, without interacting with the verifier, and this latter can also verify the transcript without any interaction.

The non-interactive argument system derived from an interactive one $\Pi = (\mathsf{Setup}, \mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Vf})$ via the Fiat–Shamir heuristic with a random oracle $\mathcal{H}$ is denoted $FS.\Pi^{\mathcal{H}} := \left(\mathsf{Setup}, \mathsf{CRSGen}, FS.\mathsf{Prove}^{\mathcal{H}}, FS.\mathsf{Vf}^{\mathcal{H}}\right)$. Note that the original interactive protocol Π may already be in the random-oracle model as in the previous section, but it is further assumed to be with a different oracle (that is not made explicit in the notation for simplicity).

Completeness. A non-interactive protocol $FS.\Pi^{\mathcal{H}}$ is said to be complete if for all $\lambda \in \mathbb{N}$, for all $pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right)$, $crs \leftarrow \mathsf{CRSGen}(pp)$, for all $(x, w) \in \mathcal{R}$, $\Pr\left[FS.\mathsf{Vf}^{\mathcal{H}}\left(crs, x, FS.\mathsf{Prove}^{\mathcal{H}}(crs, x, w)\right) = 1\right] = 1$.

As in the interactive case, the following notions of soundness and extractability can be extended to those of culpable soundness and culpable extractability.

Soundness. $FS.\Pi^{\mathcal{H}}$ satisfies $(T, q_{\mathcal{H}}, \varepsilon)$-soundness if for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ that runs in time at most $T(\lambda)$ and makes at most $q_{\mathcal{H}}(\lambda)$ queries to $\mathcal{H}$,

$$\Pr\left[FS.\mathsf{Vf}^{\mathcal{H}}(crs, x, \pi) = 1 \wedge x \notin \mathcal{L} \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right); crs \leftarrow \mathsf{CRSGen}(pp) \\ (x, \pi) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(crs) \end{array}\right] \leq \varepsilon(\lambda).$$

Extractability. $FS.\Pi^{\mathcal{H}}$ is $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathsf{Ext}}, q_{\mathcal{H}}, \varepsilon)$-extractable if for every deterministic algorithm Prove^* running in time at most $T_{\mathsf{Prove}^*}(\lambda)$ there exists an algorithm $(\mathsf{Ext}_0(Q, q) \to Q',\ \mathsf{Ext}_1(pp, x) \to w)$ such that for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$ running in time at most $T_{\mathcal{A}}(\lambda)$, if Prove^* and $\mathcal{A}$ together make at most $q_{\mathcal{H}}(\lambda)$ queries to $\mathcal{H}$,

- $*$ $$\Pr\left[(x, w) \notin \mathcal{R} : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right); crs \leftarrow \mathsf{CRSGen}(pp); Q \leftarrow \emptyset \\ (x, s) \leftarrow \mathcal{A}^{\mathsf{Ext}_0(Q, \cdot)}(crs) \\ w \leftarrow \mathsf{Ext}_1^{\mathsf{Prove}^{*\mathsf{Ext}_0(Q,\cdot)}(crs,x,s)}(crs, x) \end{array}\right] \leq \varepsilon(\lambda),$$
- $*$ the running time of Ext is at most T_{Ext} in expectation, with T_{Ext} depending on T_{Prove^*} and

$$\varepsilon_{\mathcal{A}, \mathsf{Prove}^*} := \Pr\left[b = 1 : \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^{\lambda}\right); crs \leftarrow \mathsf{CRSGen}(pp) \\ (x, s) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(crs) \\ \pi \leftarrow \mathsf{Prove}^{*\mathcal{H}(\cdot)}(crs, x, s) \\ b \leftarrow FS.\mathsf{Vf}^{\mathcal{H}(\cdot)}(crs, x, \pi) \end{array}\right].$$

Algorithm $(\mathsf{Ext}_0, \mathsf{Ext}_1)$ is given access to a Prove^* oracle which can be rewound to any step of its computation and run anew with fresh Ext_0 randomness.

Zero-Knowledge. $\Pi^{\mathcal{H}}$ is $(T_{\mathcal{A}}, T_{\mathsf{Sim}}, q_{\mathcal{H}}, \varepsilon)$-zero-knowledge if there exists an algorithm Sim running in time at most $T_{\mathsf{Sim}}(\lambda)$ such that $\mathsf{Sim}(0, Q, q) \to (h, Q)$ and $\mathsf{Sim}(1, Q, crs, x) \to (\pi, Q)$, and such that for all $\lambda \in \mathbb{N}$, for every adversary $\mathcal{A}$

that runs in time at most $T(\lambda)$ and is given oracle access to $\mathcal{H}$ and Prove,

$$\left|\Pr\left[b = 1 \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); (crs, st) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(pp) \\ b \leftarrow \mathcal{A}^{\mathcal{H}(\cdot), FS.\mathsf{Prove}^{\mathcal{H}}(crs, \cdot)}(st) \end{array}\right]\right.$$
$$\left. - \Pr\left[b = 1 \colon \begin{array}{l} pp \leftarrow \mathsf{Setup}\left(1^\lambda\right); Q \leftarrow \emptyset \\ (crs, st) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(pp) \\ b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sim}_0}(Q, \cdot), \mathcal{O}_{\mathsf{Sim}_1}(Q, crs, \cdot)}(st) \end{array}\right]\right| \leq \varepsilon(\lambda),$$

with $\mathcal{O}_{\mathsf{Sim}_0}$ an oracle that computes $(h, Q) \leftarrow \mathsf{Sim}(0, Q, q)$ on input (Q, q) and returns h, and $\mathcal{O}_{\mathsf{Sim}_1}$ an oracle that computes $(\pi, Q) \leftarrow \mathsf{Sim}(1, Q, crs, x)$ if $(x, w) \in \mathcal{R}$ and returns π, and returns $\perp$ if $(x, w) \notin \mathcal{R}$. Set Q can be considered as a state which stores all pairs (q, h) of queries and responses. The total number of random-oracle calls incured by direct $\mathcal{H}$ queries and by Prove queries from $\mathcal{A}$ can be at most $q_{\mathcal{H}}$.

3 Integer Commitments

This section recalls a scheme due to Damgård and Fujisaki which allows to commit to integers[7]. Then comes a new integer-commitment scheme with parameters smaller than those of Damgård and Fujisaki's scheme, and which are also more efficient to compute. For the version of our scheme which allows to commit to n integers, the parameters are of $O(b_{\mathbb{G}} + \log n)$ bits instead of $\Omega(n b_{\mathbb{G}} \log P)$ as with the generalized version of Damgård and Fujisaki's scheme, where $2^{b_{\mathbb{G}}}$ is an upper bound on the group order.

3.1 Damgård–Fujisaki Commitments

The Damgård–Fujisaki commitment scheme [15, 19], parameterized by a group generator G, consists of the following algorithms:

$\mathsf{Setup}\left(1^\lambda\right) \to pp$: run $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right)$, generate $h \leftarrow_\$ \mathbb{G}$ and return $(\mathbb{G}, P, h)$. Recall that these parameters are implicit inputs to all the other algorithms.

$\mathsf{KG}(pp) \to ck$: generate $\alpha \leftarrow_\$ \left[\!\left[0; 2^{b_{\mathbb{G}}+\lambda}\right]\!\right]$ ($2^{b_{\mathbb{G}}}$ is an upper bound on $\operatorname{ord}(\mathbb{G})$), compute and return $g \leftarrow h^\alpha$.

$\mathsf{Com}(g, x \in \mathbb{Z}) \to (C, d)$: generate $r \leftarrow_\$ \left[\!\left[0; 2^{b+\lambda}\right]\!\right]$, compute $C \leftarrow g^x h^r$, set $d \leftarrow (r, 1_{\mathbb{G}})$ and return (C, d).

$\mathsf{ComVf}\,(g, C, x, d) \to b \in \{0, 1\}$: parse d as $(r, \tilde{g})$. If $C = g^x h^r \tilde{g}$ and $\tilde{g}^2 = 1_{\mathbb{G}}$, return 1, else return 0.

[7] Couteau, Peters and Pointcheval [14] proved that in the case of RSA groups (with Blum integers), the security of Damgård and Fujisaki's scheme is provable under (a variant of) the RSA assumption instead of the strong RSA assumption. This also holds for our scheme. However, this result does not concern generic hidden-order groups.

Equivalently, the commitment-algorithm could simply set the decommitment information d to r, and the commitment-verification would return 1 if the equality $C^2 = \left(g^x h^d\right)^2$ holds and 0 otherwise. The squaring in the verification is due to the fact that the small-order assumption does not exclude the possibility to efficiently compute square roots of unity, and they thus relaxed the verification equation to allow for sound argument of knowledge of openings to commitments. In other words, the scheme would still be binding without the squaring in the verification equation, and the relaxation is simply an artifact to allow for sound arguments.

More precisely, suppose that the verification were not relaxed, i.e., that it would only check that $C = g^x h^d$. Two accepting transcripts (D, e_1, z_1, t_1) and (D, e_2, z_2, t_2) of a standard Schnorr-type argument of knowledge of an opening would imply that $C^{e_1-e_2} = g^{z_2-z_1} h^{t_2-t_1}$. Assuming $e_1, e_2 \in [\![0; P-1]\!]$, $e_1 \neq e_2$, and that $e_1 - e_2$ divides $z_2 - z_1$ and $t_2 - t_1$ (Damgård and Fujisaki showed that this latter event occurs with probability negligibly close to 1/2 under the assumptions on the group generator), the previous equality would imply that $\left(g^{(z_2-z_1)/(e_1-e_2)} h^{(t_2-t_1)/(e_1-e_2)} C^{-1}\right)^{e_1-e_2} = 1_{\mathbb{G}}$, and the small-order assumption would only allow to conclude that $C^2 = \left(g^{(z_2-z_1)/(e_1-e_2)} h^{(t_2-t_1)/(e_1-e_2)}\right)^2$. The trivial attack in which an adversary computes C as $g^x h^d \tilde{g}$ with $\tilde{g} \in \mathbb{G}$ such that $\tilde{g}^2 = 1_{\mathbb{G}}$ would then not be excluded by the protocol.

Properties. Damgård and Fujisaki's scheme is correct, is computationally binding under the strong-root and the μ-assumption, and is statistically hiding. Note that hiding property crucially relies on the fact that $g \in \langle h \rangle$. To guarantee the statistical hiding property of the scheme *without trusted key generation*, the party which computes g is then also required to compute a non-interactive proof that $g \in \langle h \rangle$. The commitment algorithm would then verify the proof and proceed as above if it is valid, and otherwise return $\perp$. Damgård and Fujisaki proposed to compute such a proof with a Schnorr-type protocol (made non-interactive via the Fiat–Shamir heuristic) with $\{0, 1\}$ as challenge set; i.e., given $\alpha \in \mathbb{Z}$ such that $g = h^\alpha$, the prover generates $k \leftarrow_\$ \left[\!\!\left[0; 2^{b_{\mathbb{G}}+2\lambda}\right]\!\!\right]$ and sends $f \leftarrow h^k$ to the verifier, this later chooses and sends to the prover a challenge $c \leftarrow_\$ \{0, 1\}$, the prover replies with $z \leftarrow k - c\alpha$ (computed in $\mathbb{Z}$), and the verifier accepts if and only if $h^z g^c = f$. To attain a soundness error of at most $1/P$, the proof must then be repeated at least $\lceil \log P \rceil$ times. With the Fiat–Shamir heuristic, each proof consists of (c, z), and the total proof in the public parameters then consists of $\lceil \log P \rceil \left(b_{\mathbb{G}} + 2\lambda + 2\right) = \Omega\left(b_{\mathbb{G}} \log P\right)$ bits (recall that P is super-polynomial in λ, e.g., $\lambda^{\log \lambda}$).

3.2 A new Integer-Commitment Scheme

This section introduces a novel integer-commitment scheme that is close to Damgård and Fujisaki's scheme, but with an argument (rather than a proof) of only $O(b_{\mathbb{G}})$ (with b such that $\mathrm{ord}(\mathbb{G}) \leq 2^{b_{\mathbb{G}}}$) bits in non-trusted keys, and the argument only requires a single protocol run to reach the same soundness error.

As the soundness of the protocol relies on computational assumptions on the group generator, the scheme is only computationally hiding, whereas Damgård and Fujisaki's cut-and-choose protocol is perfectly sound (the prover is not assumed to be computationally bounded) but inefficient.

Formally, let G be a group generator and let $FS.\Pi^{\mathcal{H}}$ be a Fiat–Shamir non-interactive argument system with random oracle $\mathcal{H}$ for the language $\{g \in \mathbb{G}, \ell \in \mathbb{N}^* : \exists \alpha \in \left[\!\left[0; 2^{\ell}\right]\!\right], g = h^{\alpha}\right\}$, given parameters $(\mathbb{G}, P, h, 1)$ (integer 1 is just to indicate that there is only one group element g in the word for which the proof is computed) and the empty string as CRS. The proof of the hiding property will require the protocol to satisfy culpable soundness w.r.t. the language $\sqrt{\langle h^2 \rangle}$. The scheme, parameterized by G and further denoted $\mathscr{C}$, consists of the following algorithms.

$\mathsf{Setup}\left(1^{\lambda}\right) \to pp$: run $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^{\lambda}\right)$, generate $h \leftarrow_{\$} \mathbb{G}$ and return $(\mathbb{G}, P, h)$. Recall that these parameters are implicit inputs to all the other algorithms.

$\mathsf{KG}(pp) \to ck$: generate $\alpha \leftarrow_{\$} \left[\!\left[0; 2^{b_{\mathbb{G}}+\lambda}\right]\!\right]$, compute $g \leftarrow h^{\alpha}$ and a proof $\pi \leftarrow FS.\Pi^{\mathcal{H}}.\mathsf{Prove}((\mathbb{G}, P, h, 1), (g, b_{\mathbb{G}} + \lambda), \alpha)$, and return (g, π).

$\mathsf{Com}\left((g, \pi), x \in \mathbb{Z}\right) \to (C, d)$: if $FS.\Pi^{\mathcal{H}}.\mathsf{Vf}((\mathbb{G}, P, h, 1), (g, b_{\mathbb{G}} + \lambda), \pi) = 0$, then return $\perp$; else generate $r \leftarrow_{\$} \left[\!\left[0; 2^{b_{\mathbb{G}}+\lambda}\right]\!\right]$, compute $C \leftarrow (g^x h^r)^2$, set $d \leftarrow r$ and return (C, d).

$\mathsf{ComVf}\left((g, \pi), C, x, d\right) \to b \in \{0, 1\}$: if $C^2 = \left(g^x h^d\right)^4$ return 1, else return 0.

Comparison with Damgård–Fujisaki Commitments. As for Damgård and Fujisaki's commitments, the squaring in the verification equation (compared to the computation of commitments) is again to later allow for sound arguments of knowledge of openings. The main difference compared to Damgård and Fujisaki's commitments is that commitments are computed as $(g^x h^r)^2$ instead of $g^x h^r$. It is simply due to the fact that π only guarantees that $g^2 \in \left\langle h^2 \right\rangle$, not that $g \in \langle h \rangle$, hence the power 2 in the computation of commitments to ascertain that they are hiding. However, only requiring that $g^2 \in \left\langle h^2 \right\rangle$ instead of $g \in \langle h \rangle$ is precisely what allows to have much smaller arguments that can be computed in a single protocol run.

Correctness & Security. We now prove the correctness and the security of the commitment scheme $\mathscr{C}$.

Theorem 3.1. *$\mathscr{C}$ is correct.*

Proof. The correctness of $\mathscr{C}$ immediately follows from its definition. □

Theorem 3.2. *Assuming $FS.\Pi^{\mathcal{H}}$ to be $\left(T, q_{\mathcal{H}}, \varepsilon^{\mathrm{snd}}, \sqrt{\langle h^2 \rangle}\right)$-sound, scheme $\mathscr{C}$ is $\left(T, q_{\mathcal{H}}, 2^{-\lambda+1} + \varepsilon^{\mathrm{snd}}\right)$-hiding.*

Proof. Unless the adversary can contradict the culpable soundness of $FS.\Pi^{\mathcal{H}}$, the key (g, π) it returns is such that $g^2 = h^{2\alpha}$ for some $\alpha \in \mathbb{Z}$. As the distribution of $(g^x h^r)^4 = h^{4(\alpha x + r)}$ for $r \leftarrow_{\$} \left[\!\left[0; 2^{b_{\mathbb{G}}+\lambda}\right]\!\right]$ is the same as the distribution of h^{4z} for $z \leftarrow_{\$} \left[\!\left[\alpha x \bmod \operatorname{ord}\left(h^4\right); \alpha x \bmod \operatorname{ord}\left(h^4\right) + 2^{b_{\mathbb{G}}+\lambda}\right]\!\right]$, which is at a statistical distance of at most $2^{-\lambda+1}$ from the distribution of h^{4z} for $z \leftarrow_{\$} \left[\!\left[0; \operatorname{ord}\left(h^4\right) - 1\right]\!\right]$, the claim follows. □

Theorem 3.3. *For any $T \colon \mathbb{N}^* \to \mathbb{N}^*$, scheme $\mathscr{C}$ is $\left(T, q_{\mathcal{H}}, \varepsilon^{\mathrm{strg}} + \varepsilon^{\mathrm{zk}} + 2^{-\lambda}\right)$-binding if $FS.\Pi^{\mathcal{H}}$ is $\left(T, T_{\mathsf{Sim}}, q_{\mathcal{H}}, \varepsilon^{\mathrm{zk}}\right)$-zero-knowledge and if the $(T_{\mathsf{Sim}} + O(T + b_{\mathbb{G}}),$ $\varepsilon^{\mathrm{strg}})$-strong-root assumption holds on G.*

Proof. Suppose that there exists an adversary $\mathcal{A}$ running in time T and contradicts the binding property of the scheme with probability at least ε. Consider then an algorithm $\mathcal{B}$ which runs a trapdoor setup algorithm which is exactly as $\mathsf{Setup}\left(1^\lambda\right)$ (in particular, it computes h^α for $\alpha \leftarrow_{\$} \left[\!\left[0; 2^{b_{\mathbb{G}}+\lambda}\right]\!\right]$, which requires at most $2(b_{\mathbb{G}} + \lambda + 1)$ group operations with the square-and-multiply algorithm), except that it simulates an argument of knowledge of α with the simulator of $FS.\Pi^{\mathcal{H}}$. The runtime of this trapdoor setup algorithm is then of order $T_{\mathsf{Sim}} + O(b_{\mathbb{G}} + \lambda) = T_{\mathsf{Sim}} + O(b_{\mathbb{G}})$ (since $b_{\mathbb{G}} = \Omega(\lambda)$ by assumption). Adversary $\mathcal{A}$ can then distinguish $\mathcal{B}$ from the challenger of the binding game with an advantage of at most $\varepsilon^{\mathrm{zk}}$, so $\mathcal{A}$ contradicts the binding property of the scheme with probability at least $\varepsilon - \varepsilon^{\mathrm{zk}}$ on the input of parameters returned by algorithm $\mathcal{B}$. Given two pairs of integers $(x, d), (x', d')$ and a group element C such that $x \neq x'$ and $C^2 = \left(g^x h^d\right)^4 = \left(g^{x'} h^{d'}\right)^4$, the equality $h^{4(\alpha(x-x')+d-d')} = 1_{\mathbb{G}}$ holds. Let $0 \leq \rho < \operatorname{ord}(h)$ denote the unique integer such that $\alpha = \operatorname{ord}(h) \lfloor \alpha / \operatorname{ord}(h) \rfloor + \rho$. Note that the distribution of (x, d) and (x', d') only depends on ρ as $\mathcal{A}$ is only given g (and not α) as input. Since $\lfloor \alpha / \operatorname{ord}(h) \rfloor$ is uniformly distributed over a set of size at least 2^λ, the probability that $\alpha(x - x') + d - d' = 0$ is at most $2^{-\lambda}$. Lemma 3.4 then shows that denoting by $T_{\mathcal{B}}$ the running time of $\mathcal{B}$ and setting $n := 4\left(\alpha(x - x') + d - d'\right)$, the inequality $\varepsilon - \varepsilon^{\mathrm{zk}} - 2^{-\lambda} \leq \varepsilon^{\mathrm{strg}}$ holds under the $(T_{\mathcal{B}} + O(\log n), \varepsilon^{\mathrm{strg}})$-strong-root assumption. Remark that the integers returned by $\mathcal{A}$ are necessarily less than $2^{O(T)}$ as an algorithm running in time T can return a value of at most $O(T)$ bits (the algorithm might be using an alphabet different from $\{0, 1\}$, hence the big O). Therefore, $\log n = O(T + b_{\mathbb{G}})$. Since $T_{\mathcal{B}} = T + T_{\mathsf{Sim}} + O(b_{\mathbb{G}})$, the claim follows. □

Lemma 3.4. *Consider the problem (depending on λ) of computing a value $n \in \mathbb{Z}^*$ such that $h^n = 1$ on the input of $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right)$ and of a group element $h \leftarrow_{\$} \mathbb{G}$. If there exists an algorithm $\mathcal{A}$ that solves this problem in time T with probability at least ε, then there exists an algorithm that solves the strong-root problem in time at most $T + O(\log n)$ with probability at least ε.*

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the lemma. Consider an algorithm $\mathcal{B}$ which, on the input of $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right)$ and of a group element $h \leftarrow_{\$} \mathbb{G}$, runs $\mathcal{A}$ as a subroutine on the input of $(\mathbb{G}, P)$ and h. If $\mathcal{A}$ returns an integer $n \in \mathbb{Z}^*$ such that $h^n = 1$ note that $h^{|n|+1} = h$. Algorithm $\mathcal{B}$ then computes $|n| + 1$ (which can be done in time at most $O(\log n)$) and returns $(h, |n| + 1)$. □

Argument System $\mathbf{FS.\Pi^{\mathcal{H}}}$. It only remains to provide a protocol $FS.\Pi^{\mathcal{H}}$ to argue knowledge of an integer $\alpha \in \mathbb{Z}$ such that $g^2 = h^{2\alpha}$, which is sufficient for the commitment scheme to be computationally hiding. We first give an interactive protocol Π for the language $\left\{g \in \mathbb{G}, \ell \in \mathbb{N}^* : \exists \alpha \in \left[\!\left[0; 2^{\ell}\right]\!\right], g = h^{\alpha}\right\}$ given parameters $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^{\lambda}\right)$ and that satisfies culpable soundness w.r.t. $\sqrt{\langle h^2 \rangle}$, and then apply the Fiat–Shamir heuristic to obtain $FS.\Pi^{\mathcal{H}}$.

In more detail, the (interactive) protocol Π is as follows: the prover generates $k \leftarrow_{\$} \left[\!\left[0; 2^{\ell+\lambda}P\right]\!\right]$, computes $t \leftarrow h^k$ and sends t to the verifier; the verifier chooses $c \leftarrow_{\$} [\![0; P-1]\!]$ and sends it to the prover; the prover then replies with $r \leftarrow k - c\alpha$, and the verifier accepts if and only if $h^r g^c = t$. With the Fiat–Shamir heuristic, the proof consists of (c, r), i.e., $2\lfloor \log P \rfloor + \ell + \lambda + 3$ bits. For $\ell = b_{\mathbb{G}} + \lambda$, that is $2\lfloor \log P \rfloor + b_{\mathbb{G}} + 2\lambda + 3 = O(b_{\mathbb{G}})$ bits (recall that $P \leq 2^{b_{\mathbb{G}}}$ and $b_{\mathbb{G}} = \Omega(\lambda)$).

Properties. We now show that Π is complete, statistically honest-verifier zero-knowledge and satisfies culpable extractability w.r.t. $\sqrt{\langle h^2 \rangle}$ under the assumptions introduced in Section 2.2.

Theorem 3.5. Π *is complete.*

Proof. It immediately follows from the definition of Π. □

Theorem 3.6. Π *is* $\left(O\left((\ell + \lambda + \log P) T_{\mathbb{G}}\right), 2^{-\lambda+1}\right)$*-statistically honest-verifier zero-knowledge.*

Proof. To simulate such a proof conditioned on a challenge c, it suffices to generate $k \leftarrow_{\$} \left[\!\left[0; 2^{\ell+\lambda}P\right]\!\right]$, compute $t \leftarrow h^k g^c$ and set $r \leftarrow k$. As the distribution of $\left(h^{k+c\alpha}, c, k\right)$ for $k \leftarrow_{\$} \left[\!\left[0; 2^{\ell+\lambda}P\right]\!\right]$ is at a $2^{-\lambda+1}$ statistical distance from the distribution of $\left(h^k, c, k - c\alpha\right)$ for $k \leftarrow_{\$} \left[\!\left[0; 2^{\ell+\lambda}P\right]\!\right]$, the protocol is honest-verifier $2^{-\lambda+1}$-statistically zero-knowledge. The simulator simply generates $k \leftarrow_{\$} \left[\!\left[0; 2^{\ell+\lambda}P\right]\!\right]$ and $c \leftarrow_{\$} [\![0; P-1]\!]$, computes $h^k g^c$ and returns $\left(h^k g^c, c, k\right)$. As $h^k g^c$ can be computed in $O(\ell + \lambda + \log P)$ group operations with classical sliding-window algorithms [2], the simulator runs in time $O\left((\ell + \lambda + \log P) T_{\mathbb{G}}\right)$. □

Theorem 3.7. Π *is* $\left(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, \varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu, \sqrt{\langle h^2 \rangle}\right)$*-extractable with* $T_{\mathcal{E}} := \left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*} + O(T_{\mathsf{Prove}^*} \log P)$*, and* $T_{\mathcal{A}}$ *and* T_{Prove^*} *such that* $T_{\mathcal{A}} + \left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*} + O\left(T_{\mathsf{Prove}^*} T_{\mathbb{G}} \log P + \log^2 P\right) \leq \min\left(T^{\mathrm{strg}}, T^{\mathrm{ord}}\right)$.

Proof. To prove that the protocol satisfies culpable extractability (and thus culpable soundness), notice that given two valid transcripts (t, c, r) and (t, c', r') such that $c \neq c'$, the equality $h^{r-r'} = g^{c'-c}$ holds. Assume without loss of generality that $c' > c$, and let $d := \gcd(r - r', c' - c)$. The extended Euclidean algorithm applied to $r - r'$ and $c' - c$ returns, in time $O\left(\log(r - r') \log(c' - c)\right)$, integers u and v such that $d = u(r - r') + v(c' - c)$ and $|u|, |v| \leq \max\left(|r - r'|, |c' - c|\right)/d$.

– If $d = c' - c$, then the equality $h^{r-r'} = g^{c'-c}$ implies that $\left(h^{(r-r')/(c'-c)}g^{-1}\right)^{c'-c} = 1$. The fact that $0 < c' - c < P$ and the small-order assumption therefore implies that $h^{2(r-r')/(c'-c)} = g^2$, i.e., $(r - r')/(c' - c)$ is a valid (culpable) witness.

– If $d < c' - c$, since the equality $h^{r-r'} = g^{c'-c}$ implies that $h^d = (g^u h^v)^{(c'-c)}$, $\left((g^u h^v)^{(c'-c)/d}\, h^{-1}\right)^d = 1$ although $d < c' - c < P$. The small-order assumption thus implies that $\tilde{g}^2 = 1$ for $\tilde{g} := (g^u h^v)^{(c'-c)/d}\, h^{-1}$. If $\tilde{g} = 1_{\mathbb{G}}$, then $(g^u h^v, (c' - c)/d)$ is a solution to the strong-root problem. If, $\tilde{g} \neq 1_{\mathbb{G}}$, further distinguish two sub-cases:
  - ∗ if $(c' - c)/d$ is odd, then $\tilde{g}^{(c'-c)/d} = \tilde{g}$ and therefore $h = (g^u h^v \tilde{g})^{(c'-c)/d}$, i.e., $(g^u h^v \tilde{g}, (c' - c)/d)$ is a solution to the strong-root problem
  - ∗ if $(c' - c)/d$ is even, then the low-dyadic-valuation assumption implies that $\operatorname{ord}\left((g^u h^v)^{(c'-c)/d}\right)$ is odd, which is impossible if $\operatorname{ord}(h)$ is P-rough (and necessarily odd) as $\operatorname{ord}(\tilde{g}h) = 2\operatorname{ord}(h)$ in this case.

Note that since the absolute value of integers returned by the algorithm Prove^* of a prover $(\mathcal{A}, \mathsf{Prove}^*)$ are necessarily less than $2^{O(T_{\mathsf{Prove}^*})}$, computing the witness $(r - r')/(c' - c)$ in case $d = c' - c$ can be done in time $O(T_{\mathsf{Prove}^*} \log P)$. Besides, in any of the cases in which there is a reduction to the strong-root problem, computing $(c' - c)/d$ can be done in time $O\left(\log^2 P\right)$ as $d \leq c' - c < P$, and computing $g^u h^v \tilde{g}$ can be done in $O\left(\max\left(T_{\mathsf{Prove}^*}, \log P\right)\right)$ group operations using the square-and-multiply algorithm, i.e., in time $O\left(\max\left(T_{\mathsf{Prove}^*}, \log P\right) T_{\mathbb{G}}\right)$ (recall that $T_{\mathbb{G}}$ is the bit complexity of performing group operations), after u and v have been computed in time $O\left(T_{\mathsf{Prove}^*} \log P\right)$ with the extended Euclidean algorithm. The solution to the strong-root problem can thus be computed in time $O\left(T_{\mathsf{Prove}^*} T_{\mathbb{G}} \log P + \log^2 P\right)$ given two accepting transcripts.

To obtain two valid transcripts with distinct challenges, it suffices to run the proving algorithm once, rewind it to the computation step right after it sent its first message and run it anew on fresh verifier randomness. If both runs are valid and the challenges are distinct, then return the two transcripts, otherwise restart. The expected runtime of this procedure is at most $\left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*}$ if ε denotes the success probability of the prover.

Define then an extractor $\mathcal{E}$ as an algorithm that runs the previous procedure and returns the witness $(r - r')/(c' - c)$ in case $c' - c \mid r - r'$, and otherwise returns $\perp$. The running time of $\mathcal{E}$ is then $\left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*} + O(T_{\mathsf{Prove}^*} \log P)$ in expectation.

Moreover, for T^{strg} and T^{ord} such that $T_{\mathcal{A}} + \left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*} + O\left(T_{\mathsf{Prove}^*} T_{\mathbb{G}} \log P + \log^2 P\right) \leq \min\left(T^{\text{strg}}, T^{\text{ord}}\right)$, the previous case analysis shows that a valid witness cannot be extracted with probability at $\varepsilon^{\text{ord}} + \varepsilon^{\text{strg}} + 1 - \mu$ under the $(T^{\text{strg}}, \varepsilon^{\text{strg}})$-strong-root assumption, the $\left(T^{\text{ord}}, \varepsilon^{\text{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G. The theorem thus follows. □

Arguing Knowledge of Openings. As for Damgård and Fujisaki's commitments, one can efficiently argue knowledge of openings, i.e., of integers x and r such that a given commitment C satisfies $C^2 = (g^x h^r)^4$.

The protocol imposes an upper bound of ℓ on the bit length of the witness, with ℓ being part of the (public) word. It is simply to adapt the randomness range of the prover (and of the honest-verifier zero-knowledge simulator) to ensure that the protocol remains statistically honest-verifier zero-knowledge; and ℓ can be arbitrarily large. The protocol does *not* guarantee that the largest absolute value in the extracted witness is at most ℓ bits long [8]. In technical terms, the protocol is for the relation $\left\{\left(C \in \mathbb{G}, \ell \in \mathbb{N}^*; x, r \in \left[\!\left[0; 2^\ell\right]\!\right]\right) : C^2 = (g^x h^r)^4\right\}$ that satisfies culpable extractability for the relation $\Sigma \coloneqq \left\{(C \in \mathbb{G}, \ell \in \mathbb{N}^*; x, r \in \mathbb{Z}) \colon C^2 = (g^x h^r)^4\right\}$.

More precisely, consider the problem of arguing in zero-knowledge knowledge of integers x and r such that $C^2 = (g^x h^r)^4$ and $|x|, |r| \leq 2^\ell$, for a group element C chosen by the prover and public bases h and g, and a public proof π that $g \in \sqrt{\langle h^2 \rangle}$. The prover first verifies π and aborts if it is invalid. The prover generates $y, s \leftarrow_\$ \left[\!\left[0; P2^{\ell+\lambda}\right]\!\right]$, computes and sends $D \leftarrow (g^y h^s)^2$ to the verifier. The verifier then chooses $e \leftarrow_\$ [\![0; P-1]\!]$, sends it to the prover, and this latter replies with $z \leftarrow y - ex$ and $t \leftarrow s - er$ (computed in $\mathbb{Z}$). The verifier then accepts if and only if $(g^z h^t)^2 \, C^e = D$. Further denote the protocol by $\Pi_{\mathscr{C}}$.

Theorem 3.8. *$\Pi_{\mathscr{C}}$ is complete.*

Proof. This fact immediately follows from the definition of $\Pi_{\mathscr{C}}$. □

Theorem 3.9. *$\Pi_{\mathscr{C}}$ is $\left(O\left((\ell + \lambda + \log P)\, T_{\mathbb{G}}\right), q_{\mathcal{H}}, 2^{-\lambda+1} + \varepsilon^{\mathrm{snd}}\right)$-statistically honest-verifier zero-knowledge if $\Pi^{\mathcal{H}}$ is $\left(T, q_{\mathcal{H}}, \varepsilon^{\mathrm{snd}}, \sqrt{\langle h^2 \rangle}\right)$-sound.*

Proof. Unless the adversary can contradict the culpable soundness of $\Pi^{\mathcal{H}}$, there exists $\alpha \in \mathbb{Z}$ such that $g^2 = h^{2\alpha}$. Assuming $|x|$ and $|r|$ to be at most ℓ bits long, for $y, s \leftarrow_\$ \left[\!\left[0; P2^{\ell+\lambda}\right]\!\right]$, the distribution of $\left(h^{2(\alpha y + s)}, e, y - ex, s - er\right)$ is at a statistical distance of at most $2^{-\lambda+1}$ from the distribution of $\left(h^{2(\alpha(y+ex)+(s+ex))}, e, y, s\right) = \left((g^y h^s)^2 \, C^e, e, y, s\right)$. Since $(g^y h^s)^2 \, C^e$ can be computed in $O(\ell + \lambda + \log P)$ group operations with classical sliding-window algorithms [2], the statement follows. □

Theorem 3.10. *$\Pi_{\mathscr{C}}$ is $\left(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, \left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*}, \varepsilon^{\mathrm{ext}}, \Sigma\right)$-extractable for any $T_{\mathcal{A}}$ and T_{Prove^*} such that $T_{\mathcal{A}} + b_{\mathbb{G}} \log(P) T_{\mathsf{Prove}^*} T_{\mathbb{G}} \leq \Omega\left(\min\left(T^{\mathrm{strg}}, \;\; T^{\mathrm{ord}}\right)\right)$, with $\varepsilon^{\mathrm{ext}} \coloneqq \left(1/2 - 2^{-\lambda} - (1 - \mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right) + \varepsilon^{\mathrm{zk}}_{\Pi^{\mathcal{H}}}$.*

Proof. To prove culpable extractability, note that from two accepting transcripts (D, e_1, z_1, t_1) and (D, e_2, z_2, t_2) (with the same $D \in \mathbb{G}$) such that $e_1 \neq e_2$, the equality $C^{e_2 - e_1} = (g^{z_1 - z_2} h^{t_1 - t_2})^2$ follows. If $(e_2 - e_1)$ divides $(z_1 - z_2)$ and $(t_1 - t_2)$, then

[8] To prove such statements using hidden-order groups, Lipmaa's range argument [28], corrected by Couteau, Peters and Pointcheval [14], is suitable.

the low-order assumption implies that $C = \left(g^{(z_1-z_2)/(e_2-e_1)} h^{(t_1-t_2)/(e_2-e_1)}\right)^2 \tilde{g}_C$ for some group element $\tilde{g}_C$ such that $\tilde{g}_C^2 = 1$, i.e., $((z_1-z_2)/(e_2-e_1), (t_1-t_2)/(e_2-e_1)$ is a valid witness for C. Besides, one can show that $e_1 - e_2$ does not divide $z_2 - z_1$ or $t_2 - t_1$ with probability at most $\left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)$ under the $(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}})$-strong-root assumption, the $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G, and assuming that the running time of the prover $(\mathcal{A}, \mathsf{Prove}^*)$ is such that $T_{\mathcal{A}} + \log(P) b_{\mathbb{G}} T_{\mathsf{Prove}^*} T_{\mathbb{G}} \leq \Omega\left(\min\left(T^{\mathrm{strg}},\ T^{\mathrm{ord}}\right)\right)$. This part is a special case of Lemma 4.4 and is similar to a part of the extractability proof of the protocol proposed by Damgård and Fujisaki to argue knowledge of openings to their commitments.

To obtain two valid transcripts with distinct challenges, it suffices to run the proving algorithm once, rewind it to the computation step right after it sent its first message and run it anew on fresh verifier randomness. If both runs are valid and the challenges are distinct, then return the two transcripts, otherwise restart. If a prover $(\mathcal{A}, \mathsf{Prove}^*)$ convince the verifier with probability at least ε, the expected runtime of this procedure is at most $\left(\varepsilon^2 - 1/P\right)^{-1} T_{\mathsf{Prove}^*}$. Define then an extractor $\mathcal{E}$ which first simulates a proof π that $g \in \sqrt{\langle h^2 \rangle}$ with $FS.\Pi^{\mathcal{H}}.\mathsf{Sim}$ and then runs the previous procedure. The theorem then follows.

□

Multi-Integer Commitments. The above commitments can be generalized to vectors of integers just like Damgård–Fujisaki commitments (as did Couteau, Peters and Pointcheval [14]). That is to say, the scheme can be extended to commit to several integers at once.

Formally, let G be a group generator and suppose that there exists a non-interactive argument system $FS.\Pi^{\mathcal{H}}$ with random oracle $\mathcal{H}$ for the language $\left\{g_1, \ldots, g_n \in \mathbb{G}, \ell \in \mathbb{N}^* \colon \exists \alpha_1, \ldots, \alpha_n \in \left\llbracket 0; 2^{\ell} \right\rrbracket, \forall i \in \llbracket n \rrbracket\ g_i = h^{\alpha_i} \right\}$ given parameters $(\mathbb{G}, P, h, n)$ and the empty string as CRS.

$\mathsf{Setup}\left(1^{\lambda}, n\right) \to pp$: run $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^{\lambda}\right)$, generate $h \leftarrow_{\$} \mathbb{G}$ and return $(\mathbb{G}, P, h, n)$.

$\mathsf{KG}(pp) \to ck$: generate $\alpha_i \leftarrow_{\$} \left\llbracket 0; 2^{b_{\mathbb{G}}+\lambda} \right\rrbracket$ for $i \in \llbracket n \rrbracket$, compute $g_i \leftarrow h^{\alpha_i}$ and $\pi \leftarrow FS.\Pi^{\mathcal{H}}.\mathsf{Prove}\left((\mathbb{G}, P, h, n), (\mathbf{g}, b_{\mathbb{G}} + \lambda), (\alpha_i)_{i=1}^{n}\right)$, and return $(\mathbf{g}, \pi)$.

$\mathsf{Com}\left((\mathbf{g}, \pi), x_1, \ldots, x_n \in \mathbb{Z}\right) \to (C, d)$: if $FS.\Pi^{\mathcal{H}}.\mathsf{Vf}((\mathbb{G}, P, h, n), (\mathbf{g}, b_{\mathbb{G}} + \lambda), \pi) = 0$ return $\perp$; generate $r \leftarrow_{\$} \left\llbracket 0; 2^{b_{\mathbb{G}}+\lambda} \right\rrbracket$, compute $C \leftarrow \left(\prod_{i=1}^{n} g_i^{x_i} h^r\right)^2$, set $d \leftarrow r$ and return (C, d).

$\mathsf{ComVf}\left((\mathbf{g}, \pi), C, x_1, \ldots, x_n, d\right) \to b \in \{0, 1\}$: if $C^2 = \left(\prod_i g_i^{x_i} h^d\right)^4$ return 1, else return 0.

The only missing component is an interactive protocol Π that satisfies culpable soundness w.r.t. $\left\{g_1, \ldots, g_n \in \mathbb{G} \colon \exists \alpha_1, \ldots, \alpha_n \in \mathbb{Z}, \forall i \in \llbracket n \rrbracket\, g_i^2 = h^{2\alpha_i}\right\}$. A possible solution is to run n times in parallel the protocol from the case $n = 1$ for each of the α_i values. However, the achieve an overall $2^{-\lambda}$ statistical distance from

n simulated arguments, the range of the prover's randomness in the Section–3.2 protocol must be multiplied by n so that each argument is $2^{-\lambda}n^{-1}$-zero-knowledge. A better solution is to use the protocol of Section 5.3, which results in arguments of $O(b_{\mathbb{G}} + \log n)$ bits. This should be compared to the $\Omega(nb_{\mathbb{G}} \log P)$-bit parameters of the generalized Damgård–Fujisaki commitments.

4 Succinct Inner-Product Arguments on Integers

This section gives a statistically honest-verifier zero-knowledge, logarithmic-size inner-product argument on integers committed in hidden-order groups with the scheme from Section 3.2. That is, an argument of knowledge of vectors $\mathbf{a}$ and $\mathbf{b} \in \mathbb{Z}^n$, and of a randomness $r \in \mathbb{Z}$ such that $C^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}f^r\right)^4$ and $\langle \mathbf{a}, \mathbf{b} \rangle = z$ given public bases $\mathbf{g}$ and $\mathbf{h}$, a public commitment C and a public integer z; and the bit-communication complexity of the protocol is logarithmic in of order $O(\ell + \log nb_{\mathbb{G}})$ where ℓ is an upper-bound on the bit length of the largest integer witness and $2^{b_{\mathbb{G}}}$ an upper-bound on the order of the group.

4.1 Formal Description

This section formalizes the protocol and gives precise statements as to the properties it satisfies.

Relations. The protocol is a honest-verifier zero-knowledge argument for:

$$\mathcal{R} := \left\{(C \in \mathbb{G}, z \in \mathbb{Z}, \ell \in \mathbb{N}^*; \mathbf{a}, \mathbf{b} \in \mathbb{Z}^n, r \in \mathbb{Z}) : C^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}f^r\right)^4 \wedge \langle \mathbf{a}, \mathbf{b} \rangle = z \right.$$
$$\left. \wedge \left\| \left[\mathbf{a}\ \mathbf{b}\ r\right] \right\|_\infty < 2^\ell \right\}$$

given parameters $(\mathbb{G}, P, f, n)$ with $f \in \mathbb{G}$ and $n \in \mathbb{N}^*$, and $(\mathbf{g}, \mathbf{h}, \pi_{crs}) \in \mathbb{G}^{2n} \times \{0,1\}^*$ as CRS.

The relation imposes the largest value (in absolute value) in the witness $\left[\mathbf{a}\ \mathbf{b}\ r\right]$ to be at most ℓ bits long, with ℓ being part of the (public) word. As for the argument of knowledge of openings in Section 3.2, it is again to adapt the randomness range of the prover and of the honest-verifier zero-knowledge simulator to make sure that the protocol remains statistically honest-verifier zero-knowledge; and ℓ can be arbitrarily large. However, the protocol does not necessarily return a witness with integers of at most ℓ bits in absolute value. In other words, the protocol satisfies culpable extractability w.r.t. the relation

$$\Sigma := \left\{(C \in \mathbb{G}, z \in \mathbb{Z}, \ell \in \mathbb{N}^*; \mathbf{a}, \mathbf{b} \in \mathbb{Z}^n, r \in \mathbb{Z}) : C^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}f^r\right)^4 \wedge \langle \mathbf{a}, \mathbf{b} \rangle = z\right\}.$$

The argument for $\mathcal{R}$ is actually reduced to a logarithm-size argument (given on Figure 2) for the following relation in which the inner product is also committed:

$$\mathcal{R}' := \left\{(C \in \mathbb{G}, z \in \mathbb{Z}, \ell \in \mathbb{N}^*; \mathbf{a}, \mathbf{b} \in \mathbb{Z}^n, r \in \mathbb{Z}) : C^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}e^{\langle \mathbf{a}, \mathbf{b} \rangle}f^r\right)^4 \right.$$
$$\left. \wedge \left\| \left[\mathbf{a}\ \mathbf{b}\ r\right] \right\|_\infty < 2^\ell \right\}$$

given parameters $(\mathbb{G}, P, f, n)$ with $f \in \mathbb{G}$ and $n \in \mathbb{N}^*$, and $(\mathbf{g}, \mathbf{h}, e, \pi_{crs}) \in \mathbb{G}^{2n+1} \times \{0,1\}^*$ as CRS. Again, the protocol does not guarantee that the extracted witness satisfies the bounds on its bit length – denote by Σ' the relation defined as $\mathcal{R}'$ without the restriction on the size of the witness.

During the reduction, the verifier chooses a base $e \in \langle f \rangle$ and proves to the prover that e is in $\sqrt{\langle f^2 \rangle}$, which guarantees to the prover that the commitment Ce^{2z} remains hiding. (As explained in Section 3, this precaution is not needed in groups of public prime orders.) However, since the protocol in Section 3.2 is only honest-verifier, and that the extractability of the argument system partly relies on the fact that the prover does not know a discrete-logarithm relation between e and f, the verifier must compute a non-interactive argument with a random oracle. In other words, the extractability of the argument relies on the zero-knowledge property of the protocol in Section 3.2. Moreover, the CRS of the protocol includes a proof that $\mathbf{g}$ and $\mathbf{h}$ are in $\sqrt{\langle f^2 \rangle}^n$, and the argument is only guaranteed to be honest-verifier zero-knowledge if it is indeed the case; that is, the zero-knowledge property of the argument relies on the soundness of the protocol. This mirroring in the properties of two protocols is simply due to the fact that at the beginning of the inner-product argument, the prover becomes the verifier of the protocol for $\mathbf{g}, \mathbf{h} \in \sqrt{\langle f^2 \rangle}^n$.

Main Insights. The goal is to have a protocol for $\mathcal{R}'$ in which the prover sends only $2\lceil \log n \rceil + 2$ group elements and three integers of at most $O(\ell + b_{\mathbb{G}} + \log(n)\log(P))$ bits. The main idea is to have the prover first send a constant number of commitments that depend on the witness vectors (which are in $\mathbb{Z}^n$), so that the verifier can thereafter choose *integer* linear combinations (defined by an integer x) of the witness vectors that are of length $n/2$ (to ease the explanation, further assume n to be a power of 2 in this section). These new vectors then serve as witness for a new commitment derived from the original commitment on which the proof is computed, the commitments sent by the prover and x; in bases of length $n/2$ and determined by the original bases and x. The prover and the verifier can thus recursively run the protocol with vectors of length $n/2$. After $\log n$ recursive calls, the vectors are of length 1, and the parties run a protocol that two committed integers a and b satisfy $ab = z$ for a public z.

In more detail, given $\mathbf{a}, \mathbf{b} \in \mathbb{Z}^n$ and $r \in \mathbb{Z}$ such that $C^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}e^{\langle \mathbf{a}, \mathbf{b} \rangle}f^r\right)^4$, the prover first sends commitments $U \leftarrow \left({\mathbf{g}_1}^{\mathbf{a}_2}{\mathbf{h}_2}^{\mathbf{b}_1}e^{\langle \mathbf{a}_2, \mathbf{b}_1 \rangle}f^{s_u}\right)^2$ and $V \leftarrow \left({\mathbf{g}_2}^{\mathbf{a}_1}{\mathbf{h}_1}^{\mathbf{b}_2} e^{\langle \mathbf{a}_1, \mathbf{b}_2 \rangle}f^{s_v}\right)^2$, for s_u and s_v with uniform distribution over an integer set large enough for the commitments to be hiding. The verifier chooses $x \leftarrow_{\$} [\![0; P-1]\!]$, sends it to the prover, and this latter computes $\mathbf{a}' \leftarrow \mathbf{a}_1 + x\mathbf{a}_2$, $\mathbf{b}' \leftarrow x\mathbf{b}_1 + \mathbf{b}_2$ and $t \leftarrow s_v + rx + s_u x^2$. Remark that all these operation are performed in $\mathbb{Z}$ and do not require to invert any integer. Now note that

$$\left((\mathbf{g}_1^x \circ \mathbf{g}_2)^{\mathbf{a}'} (\mathbf{h}_1 \circ \mathbf{h}_2^x)^{\mathbf{b}'} e^{\langle \mathbf{a}', \mathbf{b}' \rangle} f^t\right)^4 = \left(U^{x^2} C^x V\right)^2,$$

which means that the prover and verifier can run the protocol again with $\mathbf{g}_1^x \circ \mathbf{g}_2$ and $\mathbf{h}_1 \circ \mathbf{h}_2^x$ as bases and $\mathbf{a}'$ and $\mathbf{b}'$ (all of size $n/2$ instead of n) as witness for $U^{x^2} C^x V$.

To understand how a witness consisting of integer vectors can be extracted, suppose that one can obtain three transcripts $\left(U, V, x_j, \mathbf{a}'_j, \mathbf{b}'_j, t'_j\right)_{i=1}^{3}$ such that

$$\left((\mathbf{g}_1{}^{x_j} \circ \mathbf{g}_2)^{\mathbf{a}'_j} \, (\mathbf{h}_1 \circ \mathbf{h}_2{}^{x_j})^{\mathbf{b}'_j} \, e^{\left\langle \mathbf{a}'_j, \mathbf{b}'_j \right\rangle} f^{t_j}\right)^4 = \left(U^{x_j^2} C^{x_j} V\right)^2$$

for all $j \in [\![3]\!]$. The goal is to find a representation of C in the bases $\mathbf{g}$, $\mathbf{h}$, e and f. To do so, consider the linear system:

$$\mathbf{X} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ for } \mathbf{X} := \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix} \text{ and indeterminate } \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix}.$$

It does not necessarily have a solution in $\mathbb{Z}^3$ (and this is the first major difference with Bulletproofs in groups with public prime orders). However, denoting by $\operatorname{adj}(\mathbf{X})$ the adjugate matrix of $\mathbf{X}$ (which is in $\mathbb{Z}^{3\times 3}$), the column vector

$$\nu_C := \operatorname{adj}(\mathbf{X}) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ satisfies } \mathbf{X}\nu_C = \mathbf{X} \operatorname{adj}(\mathbf{X}) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \det(\mathbf{X}) \\ 0 \end{bmatrix}$$

since $\mathbf{X} \operatorname{adj}(\mathbf{X}) = \det(\mathbf{X})\mathbf{I}_3$. Therefore, via linear combinations with coefficient determined by ν_C, one can obtain $\mathbf{a}_C, \mathbf{b}_C \in \mathbb{Z}^n$ and $z_C, r_C \in \mathbb{Z}$ such that $U^{2 \det \mathbf{X}} = \left(\mathbf{g}^{\mathbf{a}_C} \mathbf{h}^{\mathbf{b}_C} e^{z_C} f^{r_C}\right)^4$. If the challenges x_1, x_2, x_3 are pairwise distinct, then $\det \mathbf{X} \neq 0$, and Lemma 4.4 shows that under the assumptions on the group generator, $2 \det \mathbf{X}$ must divide (with overwhelming probability) $4z_C$, $4r_C$ and each of the components of $4\mathbf{a}_C$, $4\mathbf{b}_C$. Therefore, up to a relabeling of $2\mathbf{a}_C / \det \mathbf{X}$ and so on, one can extract $\mathbf{a}_C, \mathbf{b}_C \in \mathbb{Z}^n$ and $z_C, r_C \in \mathbb{Z}$ such that $U = \left(\mathbf{g}^{\mathbf{a}_C} \mathbf{h}^{\mathbf{b}_C} e^{z_C} f^{r_C}\right)^2 \tilde{g}_C$ for $\tilde{g}_C \in \mathbb{G}$ that satisfies $\tilde{g}_C^2 = 1_{\mathbb{G}}$.

Nonetheless, it is not yet certain that $z_C = \langle \mathbf{a}_c, \mathbf{b}_C \rangle$. To guarantee it, it suffices to extract similar representations for U and V, and replacing U, C and V by those representations in the equality $\left(\left(\mathbf{g}_1^x \circ \mathbf{g}_2\right)^{\mathbf{a}'} \left(\mathbf{h}_1 \circ \mathbf{h}_2^x\right)^{\mathbf{b}'} e^{\langle \mathbf{a}', \mathbf{b}' \rangle} f^t\right)^4 = \left(U^{x^2} C^x V\right)^2$ for any $x \in \{x_1, x_2, x_3\}$. This leads to a discrete-logarithm relation $1_{\mathbb{G}} = \mathbf{g}_1{}^{p_{\mathbf{g}_1}(x)} \mathbf{g}_2{}^{p_{\mathbf{g}_2}(x)} \mathbf{h}_1{}^{p_{\mathbf{h}_1}(x)} \mathbf{h}_2{}^{p_{\mathbf{h}_2}(x)} e^{p_e(x)} f^{p_f(x)}$ with $p_{\mathbf{g}_1}, p_{\mathbf{g}_2}, p_{\mathbf{h}_1}, p_{\mathbf{h}_2}, p_e, p_f$ polynomials in $\mathbb{Z}[x]$ of degree at most 2. Lemma 4.5 essentially states that it is hard to find discrete-logarithm relations in the subgroup generated by a group element $f \leftarrow_{\$} \mathbb{G}$ (this is the second main difference with Bulletproofs in groups with public prime orders). It thus implies that if the bases are all in $\langle f \rangle$ with exponents chosen uniformly at random over a large integer set, these polynomials must all be zero (with overwhelming probability) when evaluated at x; and $p_{\mathbf{g}_1}, p_{\mathbf{h}_2}$ and p_e together lead to an integer polynomial of degree 4, with leading coefficient $z_C - \langle \mathbf{a}_c, \mathbf{b}_C \rangle$, that must be nil when evaluated at x. Therefore, starting with five accepting transcripts instead of three entails that this polynomial of degree 4 must be nil and thus $z_C = \langle \mathbf{a}_c, \mathbf{b}_C \rangle$, i.e., $\mathbf{a}_C, \mathbf{b}_C \in \mathbb{Z}^n, r_C \in \mathbb{Z}$ is a valid witness for C.

As for the zero-knowledge property of the scheme, the ranges of s_u and s_v at each of the $\log n$ recursion step are chosen so that the statistical distance of (U, V) to a pair of uniform values in $\left\langle f^2 \right\rangle$ is at most $\left(\log(n)2^\lambda\right)^{-1}$. It then remains to compute an upper-bound on the bit length of the witness at the lass step of the protocol so that the randomness of the prover can be chosen from a set of which the bit length is λ times larger. The calculation is detailed in the proof of Theorem 4.2.

Protocol Algorithms. The argument system for relation $\mathcal{R}$ is further denoted Π. It uses as building blocks a group generator G and the Fiat–Shamir non-interactive variant $FS.\tilde{\Pi}^{\mathcal{H}}$ with a random oracle $\mathcal{H}$ of a protocol $\tilde{\Pi}$ for the language

$$\left\{(\mathbf{g}, \mathbf{h}) \in \mathbb{G}^{2n} \colon \exists \alpha, \beta \in \mathbb{Z}^{2n}, \forall i \in [\![n]\!]\, g_i = f^{\alpha_i} \wedge h_i = f^{\beta_i}\right\}$$

given parameters $(\mathbb{G}, P, f, 2n)$ and the empty string as CRS. $\tilde{\Pi}^{\mathcal{H}}$ is later assumed to satisfy culpable soundness w.r.t. the language

$$\left\{(\mathbf{g}, \mathbf{h}) \in \mathbb{G}^{2n} \colon \exists \alpha, \beta \in \mathbb{Z}^{2n}, \forall i \in [\![n]\!]\, g_i^2 = f^{2\alpha_i} \wedge h_i = f^{2\beta_i}\right\}.$$

The protocol algorithms are then as follows:

- $\Pi.\mathsf{Setup}\left(1^\lambda, n \in \mathbb{N}^*\right)$ runs $(\mathbb{G}, P') \leftarrow \mathsf{G}\left(1^\lambda\right)$, computes $P := \left\lfloor P'^{1/3} \right\rfloor$ (the power 1/3 is to ensure extractability under the assumptions on the group generator), generates $f \leftarrow_\$ \mathbb{G}$ and returns $pp \leftarrow (\mathbb{G}, P, n, f)$ as public parameters.
- $\Pi.\mathsf{CRSGen}(pp)$ generates $\alpha_i, \beta_i \leftarrow_\$ \left[\!\left[0; 2^{b_{\mathbb{G}}+2\lambda}\right]\!\right]$ for $i \in [\![n]\!]$, computes $g_i \leftarrow f^{\alpha_i}$, $h_i \leftarrow f^{\beta_i}$ and $\pi_{crs} \leftarrow FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Prove}\left((\mathbb{G}, P, f, 2n), (\mathbf{g}, \mathbf{h}), \alpha, \beta\right)$, and returns $(\mathbf{g}, \mathbf{h}, \pi_{crs})$.
- $\Pi.\mathsf{Prove}$ and $\Pi.\mathsf{Vf}$ are as on Figure 1. They run as sub-routines the proving and verification algorithms of a protocol Π' for relation $\mathcal{R}'$.
  - ○ $\tilde{\Pi}.\mathsf{Setup}\left(1^\lambda, n \in \mathbb{N}^*\right)$ is the same as $\Pi.\mathsf{Setup}\left(1^\lambda, n \in \mathbb{N}^*\right)$.
  - ○ $\tilde{\Pi}.\mathsf{CRSGen}(pp)$ computes $\mathbf{g}$ and $\mathbf{h}$ as $\Pi.\mathsf{CRSGen}(pp)$, and also $e \leftarrow f^\gamma$ for $\gamma \leftarrow_\$ \left[\!\left[0; 2^{b_{\mathbb{G}}+2\lambda}\right]\!\right]$. It then computes a proof $\pi_{crs} \leftarrow FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Prove}\left((\mathbb{G}, P, f, 2n+1), (\mathbf{g}, \mathbf{h}, e), \alpha, \beta, \gamma\right)$ that the bases g_i, h_i for $i \in [\![n]\!]$ and e are in $\sqrt{\langle f^2 \rangle}$, and eventually returns the tuple $(\mathbf{g}, \mathbf{h}, e, \pi_{crs})$
  - ○ $\Pi'.\mathsf{Prove}$ and $\Pi'.\mathsf{Vf}$ are as on Figure 2, except that $\Pi'.\mathsf{Prove}$ first verifies the CRS and aborts if it is invalid. In the presentation on Figure 2, the CRS does not include the proof that it is well-formed and the prover does not perform any preliminary verification (i.e., the CRS is assumed to be honestly generated). The reason is that $\Pi.\mathsf{Prove}$ already does this verification; if Π' were to be used as a stand-alone protocol, those verification would be necessary. $\Pi'.\mathsf{Prove}$ and $\Pi'.\mathsf{Vf}$ additionally take as input

a variable i which keeps track of the recursion depth during the protocol execution to adjust the randomness of the prover.

$\mathcal{P}\ (n, f, \mathbf{g}, \mathbf{h}, \pi_{crs}, C, z, \ell; \mathbf{a}, \mathbf{b}, r)$ $\mathcal{V}\ (n, f, \mathbf{g}, \mathbf{h}, \pi_{crs}, C, z, \ell)$

$$C^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} f^r\right)^4 \wedge \langle \mathbf{a}, \mathbf{b} \rangle = z \wedge \left\| \left[\mathbf{a}\ \mathbf{b}\ r \right] \right\|_\infty < 2^\ell$$

$\alpha \leftarrow_\$ \left[\!\left[0; 2^{b+2\lambda} \right]\!\right]$; $e \leftarrow f^\alpha$

$\pi \leftarrow FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Prove}((\mathbb{G}, P, f, 1), e, \alpha)$

$\xleftarrow{e, \pi}$

if $FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Vf}\ ((\mathbb{G}, P, f, 2n), (\mathbf{g}, \mathbf{h}), \pi_{crs}) = 0$
then return $\perp$

if $FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Vf}((\mathbb{G}, P, f, 1), e, \pi) = 0$
then return $\perp$
run the protocol on Figure 2 on input $\left(1, n, f, \mathbf{g}, \mathbf{h}, e, C_1 := Ce^{2z}, \ell; \mathbf{a}, \mathbf{b}, r\right)$

Fig. 1. Inner-Product Argument on Integers.

Prover-Communication Complexity. Throughout the protocol, the prover sends $2n' + 2$ group elements (with $n' = \lceil \log n \rceil$), two integers ($a'$ and b') less than $2^\ell P^{n'}$ in absolute value and an integer (u) less than $\left(2n' 2^{b_{\mathbb{G}}+\lambda} P^{n'+3} + 2^\ell (P-1)^{n'+2}\right) \left(1 + 2^\lambda\right)$ in absolute value. The bit communication complexity of the prover is then of order $O\left(\ell + \log(n)(b_{\mathbb{G}} + \log P) + \lambda + \max\left(\log\log n + b_{\mathbb{G}} + \lambda, \ell\right)\right)$. Since $\log P \leq b_{\mathbb{G}} = \Omega(\lambda)$, that is $O\left(\ell + \log(n) b_{\mathbb{G}} + \max\left(\log\log n + b_{\mathbb{G}}, \ell\right)\right)$, or even $O\left(\ell + \log(n) b_{\mathbb{G}}\right)$ bits (n is here assumed to be greater than 1).

Verification via a Single Multi-Exponentiation. As described on Figure 2, the verifier computes a new commitment $U_i^{x_i^2} C_i^{x_i} V_i$, and new vectors $\mathbf{g}_1^{x_i} \circ \mathbf{g}_2$ and $\mathbf{h}_1 \circ \mathbf{h}_2^{x_i}$ at each recursion step i. In total, the verifier then has to compute $n' := \lceil \log n \rceil$ 3-exponentiation with exponents less than P^2 and two $\left\lceil n 2^{-i} \right\rceil$-exponentiations with exponents less than P for $i = 0, \ldots, n' - 1$. At the last stage of the protocol, the verifier also has to check that $\left(g^{x_{n'+1} a'} h^{x_{n'+1} b'} e^{a'b'} f^u\right)^4 = \left(C_{n'+1}^{x_{n'+1}^2} \Gamma^{x_{n'+1}} \Delta\right)^2$, i.e., a 7-exponentiation with exponents (in absolute value) less than the bit length of the largest exponent.

Alternatively, the verifier could simply generate the challenges after receiving the U_i and V_i values, delay its verification to the last stage of the protocol and then do a single multi-exponentiation. As shown below, this multi-exponentiation is a $(2n + 2n' + 5)$-exponentiation, which results in computational savings in practice since computing a k-exponentiation with ℓ-bit exponents requires ℓ group operations with a pre-computed table of 2^k group elements following classical sliding-window methods [2], which is much faster than computing

$\mathcal{P}\ (i, n, f, \mathbf{g}, \mathbf{h}, e, C_i, \ell; \mathbf{a}, \mathbf{b}, r)$ $\qquad\qquad$ $\mathcal{V}\ (i, n, f, \mathbf{g}, \mathbf{h}, e, C_i, \ell)$

$$C_i^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}e^{\langle \mathbf{a},\mathbf{b}\rangle}f^r\right)^4 \wedge \left\|\left[\mathbf{a}\ \mathbf{b}\ r\right]\right\|_\infty < 2^\ell$$

if $n = 1$

$\alpha, \beta \leftarrow_\$ \left[\!\left[0; 2^{\ell+\lambda}P^i\right]\!\right]$
$s \leftarrow_\$ \left[\!\left[0; 2i - 12^{b_{\mathbb{G}}+\lambda}\right]\!\right]$
$t \leftarrow_\$ \left[\!\left[0; 2i - 12^{b_{\mathbb{G}}+2\lambda}P^{i+2} + 2^{\ell+\lambda}(P-1)^{i+1}\right]\!\right]$
Replace $i - 1$ by 1 if $i = 1$
$\Gamma \leftarrow \left(g^\alpha h^\beta e^{\alpha b + a\beta} f^s\right)^2$
$\Delta \leftarrow \left(e^{\alpha\beta} f^t\right)^2$

$\xrightarrow{\Gamma,\Delta}$

$x_i \leftarrow_\$ [\![0; P-1]\!]$

$\xleftarrow{x_i}$

$a' \leftarrow \alpha + a x_i$
$b' \leftarrow \beta + b x_i$
$u \leftarrow t + s x_i + r x_i^2$

$\xrightarrow{a',b',u}$

$\left(g^{x_i a'} h^{x_i b'} e^{a' b'} f^u\right)^4 \overset{?}{=} \left(C_i^{x_i}\Gamma^{x_i}\Delta\right)^2$

else

$s_u, s_v \leftarrow_\$ \left[\!\left[0; 2\left(\lceil \log n \rceil + i - 1\right) 2^{b_{\mathbb{G}}+\lambda}\right]\!\right]$
$U_i \leftarrow \left({\mathbf{g}_1}^{\mathbf{a}_2} {\mathbf{h}_2}^{\mathbf{b}_1} e^{\langle \mathbf{a}_2, \mathbf{b}_1\rangle} f^{s_u}\right)^2$
$V_i \leftarrow \left({\mathbf{g}_2}^{\mathbf{a}_1} {\mathbf{h}_1}^{\mathbf{b}_2} e^{\langle \mathbf{a}_1, \mathbf{b}_2\rangle} f^{s_v}\right)^2$

$\xrightarrow{U_i,V_i}$

$x_i \leftarrow_\$ [\![0; P-1]\!]$

$\xleftarrow{x_i}$

$\mathbf{a}' \leftarrow \mathbf{a}_1 + x_i\mathbf{a}_2$
$\mathbf{b}' \leftarrow x_i\mathbf{b}_1 + \mathbf{b}_2$
$t \leftarrow s_v + r x_i + s_u x_i^2$

recurse on $\left(i+1,\ \lceil n/2 \rceil,\ f,\ \mathbf{g}_1^{x_i} \circ \mathbf{g}_2,\ \mathbf{h}_1 \circ \mathbf{h}_2^{x_i},\ e,\ C_{i+1} \coloneqq U_i^{x_i^2} C_i^{x_i} V_i,\ \ell; \mathbf{a}', \mathbf{b}', t\right)$

Fig. 2. Argument for Relation $\mathcal{R}'$.

k separate single exponentiations with ℓ-bit exponents (which requires $k\ell$ group operations with a single group element in memory) and multiplying the result. Of course, if n is large, then the pre-computation might be prohibitively long with the standard multi-exponentiation method, in which case one would rather split the multi-exponentiation in small batches of appropriate size. In any case, delaying the verification until the last step already has the benefit of eliminating latency in the verification.

To reduce the verification to a single multi-exponentiation, the commitment at the last stage and the bases g and h must be expressed in terms of the challenges x_i and of the initial vectors $\mathbf{g}$ and $\mathbf{h}$ alone.

Ultimate Commitment. Given that $C_{i+1} := U_i^{x_i^2} C_i^{x_i} V_i$ for all $i \in [\![n']\!]$, the commitment $C_{n'+1}$ at the last step of the protocol is equal to

$$U_{n'}^{x_{n'}^2} \prod_{i=1}^{n'-1} U_i^{x_i^2 x_{i+1} \cdots x_{n'}} C^{x_1 \cdots x_{n'}} \prod_{i=1}^{n'-1} V_i^{x_{i+1} \cdots x_{n'}} V_{n'}.$$

Expression for g and h. It now remains to express the bases g and h at the last step of the recursion in terms of g_i, h_i for $i \in [\![n]\!]$ and x_j for $j \in [\![n']\!]$.

First assume n to be a power of 2 and let $n' := \log n$. In this case,

$$g = \prod_{i=1}^{n} g_i^{\prod_{j \in S_i} x_j} \quad \text{and} \quad h = \prod_{i=1}^{n} h_i^{\prod_{j \in [\![n]\!] \setminus S_i} x_j}$$

with

$$S_i := \{ j \in [\![n']\!] : n' + 1 - j\text{th bit of } i - 1 \text{ is } 0 \}.$$

In case n is not a power of 2, finding an explicit expression for the exponents x_j to which g_i is raised is much more intricate. For instance, in case $n = 5$, $g = g_1^{x_1 x_2 x_3} g_2^{x_1 x_3} g_3^{x_2 x_3} g_4^{x_3} g_5$. The above expression is no longer true since, for example, the exponent g_5 is 1 although the first and second bits of 4 are nil and the exponent of g_4 is x_3 even though the first bit of 3 is 1. Appendix A.1 then gives an explicit expression in case n is not a power of 2.

Another possibility is to expand, $\mathbf{g}$ and $\mathbf{h}$ to vectors $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ of size $2^{n'}$ by inserting $1_{\mathbb{G}}$ at specific positions so that the result of the folding procedure applied $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ are the same as the result of the procedure applied to $\mathbf{g}$ and $\mathbf{h}$ (i.e., g and h). This reduces the multi-exponentiation to the case in which the size of the vectors is a power of 2. We give an algorithm to do so in Appendix A.1 and prove its correctness.

Verification Efficiency. In case n is a power of 2 (the previous paragraph shows that the problem can always be reduced to this case), the verifier then only has to check that

$$\left(\prod_{i=1}^{n} g_i^{\prod_{j \in S_i} x_j} \right)^{4 x_{n'+1} a'} \left(\prod_{i=1}^{n} h_i^{\prod_{j \in [\![n]\!] \setminus S_i} x_j} \right)^{4 x_{n'+1} b'} e^{4a'b'} f^{4u}$$

$$= \left(U_{n'}^{x_{n'}} \prod_{i=1}^{n'-1} U_i^{x_i x_{i+1} \cdots x_{n'}} C^{x_1 \cdots x_{n'}} \prod_{i=1}^{n'-1} V_i^{x_{i+1} \cdots x_{n'}} V_{n'} \right)^{2 x_{n'+1}^2} \Gamma^{2 x_{n'+1}} \Delta^2,$$

i.e., do a $(2n + 2n' + 5)$-exponentiation with exponents (in absolute value) less than

$$4 \max\left(2^{\ell} P^{2n'+1}, \overbrace{2^{2\ell} P^{2n'}}^{|a'b'|<}, \overbrace{\left(2n' 2^{b_{\mathbb{G}}+\lambda} P^{n'+1} + 2^{\ell} (P-1)^{n'+2}\right)\left(1 + 2^{\lambda}\right)}^{|u|<}\right).$$

Verification thus requires $O(\ell + b_{\mathbb{G}} + \log(n)\log(P))$ group operations ($n \geq 2$).

4.2 Completeness and Security

This section proves that Π is complete, honest-verifier zero-knowledge and extractable. The proof of extractability of Π is based on the proof of extractability of Π' and on Lemma 4.5. The proof of extractability of Π' is itself based on Lemma 4.5 and Lemma 4.4, and Lemma 4.4 relies on Lemma 4.3.

Theorem 4.1 (Completeness). Π *is complete.*

Proof. The completeness of the protocol immediately follows from the completeness of protocol $\Pi^{\mathcal{H}}$, from the fact that each step $i \in [\![n']\!]$,

$$\begin{aligned}
&\left(\left(\mathbf{g}_1^{x_i} \circ \mathbf{g}_2\right)^{\mathbf{a}'} \left(\mathbf{h}_1 \circ \mathbf{h}_2^{x_i}\right)^{\mathbf{b}'} e^{\langle \mathbf{a}', \mathbf{b}' \rangle} f^{t}\right)^4 \\
&= \left(\mathbf{g}_1^{x_i} \circ \mathbf{g}_2\right)^{4(\mathbf{a}_1 + x_i \mathbf{a}_2)} \left(\mathbf{h}_1 \circ \mathbf{h}_2^{x_i}\right)^{4(x_i \mathbf{b}_1 + \mathbf{b}_2)} e^{4\left(x^2 \langle \mathbf{a}_2, \mathbf{b}_1 \rangle + x \langle \mathbf{a}, \mathbf{b} \rangle + \langle \mathbf{a}_1, \mathbf{b}_2 \rangle\right)} f^{4(s_v + r x_i + s_u x_i^2)} \\
&= \mathbf{g}_1{}^{4x_i(\mathbf{a}_1 + x_i \mathbf{a}_2)} \mathbf{g}_2{}^{4(\mathbf{a}_1 + x_i \mathbf{a}_2)} \mathbf{h}_1{}^{4(x_i \mathbf{b}_1 + \mathbf{b}_2)} \mathbf{h}_2{}^{4x_i(x_i \mathbf{b}_1 + \mathbf{b}_2)} \\
&= \left(U_i^{x_i^2} C_i^{x_i} V_i\right)^2,
\end{aligned}$$

and from the completeness of the last step of the protocol (i.e., $i = n' + 1$). The second equality stem from the fact that for any two vectors $\mathbf{G}, \mathbf{H} \in \mathbb{G}^n$ and any vector $\mathbf{k} \in \mathbb{Z}^n$, $(\mathbf{G} \circ \mathbf{H})^{\mathbf{k}} = \mathbf{G}^{\mathbf{k}} \mathbf{F}^{\mathbf{k}}$. □

Theorem 4.2 (Honest-Verifier Zero-Knowledge). *If* $\tilde{\Pi}$ *is* $\left(T_{\tilde{\Pi}}, q_{\mathcal{H}}, \varepsilon_{\tilde{\Pi}}^{\mathrm{snd}}\right)$-*sound, then protocol* Π *is* $\left(T_{\tilde{\Pi}}, O\left(\left(\ell + b_{\mathbb{G}} + \log(P)\log(n)\right) T_{\mathbb{G}}\right), q_{\mathcal{H}}, 2^{-\lambda+2} + \varepsilon_{\tilde{\Pi}}^{\mathrm{snd}}\right)$-*honest-verifier zero-knowledge.*

Proof. If the initial length n of the vectors is at least 2, note that at each step $i \in [\![n']\!]$, the length of the vectors is $n_i := \left\lceil n 2^{-i+1} \right\rceil$, so $\lceil \log n_i \rceil + i - 1 = n'$. Since $e^2 \in \left\langle f^2 \right\rangle$ (the verifier is honest), and since the group elements in the parameters are also in $\left\langle f^2 \right\rangle$ unless the adversary can contradict the soundness of $\tilde{\Pi}$, the pair (U_i, V_i) is at a $2^{-\lambda+1}/n'$ statistical distance from a pair $\left(f^{2s_u}, f^{2s_v}\right)$ for $s_u, s_v \leftarrow_{\$} \left[\!\left[0; 2n' 2^{b_{\mathbb{G}}+\lambda}\right]\!\right]$, and the tuple $(U_i, V_i)_{i=1}^{n'}$ is thus at a statistical distance of at most $2^{-\lambda+1}$ from a $2n'$-tuple with components generated as f^{2s} for $s \leftarrow_{\$} \left[\!\left[0; 2n' 2^{b_{\mathbb{G}}+\lambda}\right]\!\right]$.

At the last step of the protocol (i.e., for $i = n' + 1$), note that $|a|, |b| \leq 2^{\ell} P^{n'+1}$ and that the absolute value of the randomness r in the commitment $C_{n'+1}$ is less than $2n'2^{b_{\mathbb{G}}+\lambda}P^{n'+1}+2^{\ell}\left(P-1\right)^{n'}$. Indeed, if $n = 1$ the statement is immediate and if $n \geq 2$, at the first step of the protocol $i = 1$, $\max \mathbf{a}' = \max \mathbf{a}_1 + x_1 \max \mathbf{a}_2 \leq 2^{\ell}P$. If $\min \mathbf{a}_2 < 0$, then $\min \mathbf{a}' = \min \mathbf{a}_1 + x_1 \min \mathbf{a}_2 \geq -2^{\ell}P$, and if $\min \mathbf{a}_2 \geq 0$, then $\min \mathbf{a}' = \min \mathbf{a}_1 \geq -2^{\ell} \geq -2^{\ell}P$; and similarly for $\mathbf{b}'$. A simple induction then shows that $|a|, |b| \leq 2^{\ell}P^{n'+1}$. Besides, the randomness of the commitment C_1 is at most 2^{ℓ} in absolute value, and under the assumption that the randomness in C_i is at most $2n'2^{b_{\mathbb{G}}+\lambda}P^{i} + 2^{\ell}\left(P-1\right)^{i-1}$ in absolute value for $i \in [\![n']\!]$, the randomness in C_{i+1} is less than

$$\begin{aligned}
&2n'2^{b_{\mathbb{G}}+\lambda} + \left(2n'2^{b_{\mathbb{G}}+\lambda}P^{i} + 2^{\ell}\left(P-1\right)^{i-1}\right)(P-1) + 2n'2^{b_{\mathbb{G}}+\lambda}(P-1)^2\\
&= 2n'2^{b_{\mathbb{G}}+\lambda}\left(1 + P^{i}(P-1) + (P-1)^2\right) + 2^{\ell}\left(P-1\right)^{i}\\
&\leq 2n'2^{b_{\mathbb{G}}+\lambda}\left(P^{i}(P-1) + P^2\right) + 2^{\ell}\left(P-1\right)^{i}\\
&\leq 2n'2^{b_{\mathbb{G}}+\lambda}P^{i+1} + 2^{\ell}\left(P-1\right)^{i}.
\end{aligned}$$

Moreover, it is also greater than $-2^{\ell}(P-1)^{i}$ since the randomness s_u and s_v are always positive. The statement then holds for all $i \in [\![n'+1]\!]$.

It follows that at the last step of the protocol, (assuming $n' \geq 1$,)

$$\begin{aligned}
sx_{n'+1} + rx_{n'+1}^2 &\leq 2n'2^{b_{\mathbb{G}}+\lambda} + \left(2n'2^{b_{\mathbb{G}}+\lambda}P^{n'+1} + 2^{\ell}\left(P-1\right)^{n'}\right)(P-1)^2\\
&\leq 2n'2^{b_{\mathbb{G}}+\lambda}\left(1 + P^{n'+1}(P-1)^2\right) + 2^{\ell}(P-1)^{n'+2}\\
&\leq 2n'2^{b_{\mathbb{G}}+\lambda}P^{n'+3} + 2^{\ell}(P-1)^{n'+2}.
\end{aligned}$$

Consequently, for $\alpha, \beta \leftarrow_{\$} \left[\!\!\left[0; 2^{\ell+\lambda}P^{n'+1}\right]\!\!\right]$, $s \leftarrow_{\$} \left[\!\!\left[0; 2\max(n', 1)2^{b_{\mathbb{G}}+\lambda}\right]\!\!\right]$ and $t \leftarrow_{\$} [\![0;\ 2\max(n', 1)2^{b_{\mathbb{G}}+2\lambda}P^{n'+3} +\ 2^{\ell+\lambda}\left(P-1\right)^{n'+2}\!\!\left.\right]\!\!]$, the distribution of

$$\left(\overbrace{\left(g^{\alpha}h^{\beta}e^{\alpha b+a\beta}f^{s}\right)^2}^{\Gamma}, \overbrace{\left(g^{x_{n'+1}a'}h^{x_{n'+1}b'}e^{a'b'}f^{u}\right)^2 C_{n'+1}^{-x_{n'+1}^2}\Gamma^{-x_{n'+1}}}^{\Delta=\left(e^{\alpha\beta}f^{t}\right)^2}, x_{n'+1}, \overbrace{\alpha + ax_{n'+1}}^{a'}, \overbrace{\beta + bx_{n'+1}}^{b'}, \overbrace{t + sx_{n'+1} + rx_{n'+1}^2}^{u}\right)$$

is at a statistical distance of at most $2^{-\lambda}$ from the distribution of

$$\left(\left(g^{\alpha}h^{\beta}e^{\alpha b+a\beta}f^{s}\right)^2, \left(g^{x_{n'+1}\alpha}h^{x_{n'+1}\beta}e^{\alpha\beta}f^{t}\right)^2 C_{n'+1}^{-x_{n'+1}^2}\Gamma^{-x_{n'+1}}, x_{n'+1}, \alpha, \beta, t\right),$$

which is itself at a statistical distance of at most $2^{-\lambda}$ from the distribution of

$$\left(f^{2s}, \left(g^{x_{n'+1}\alpha}h^{x_{n'+1}\beta}e^{\alpha\beta}f^{t}\right)^2 C_{n'+1}^{-x_{n'+1}^2}f^{-2sx_{n'+1}}, x_{n'+1}, \alpha, \beta, t\right).$$

Recall also that

$$C_{n'+1} = U_{n'}^{x_{n'}^2}\prod_{i=1}^{n'-1}U_i^{x_i^2x_{i+1}\cdots x_{n'}}C^{x_1\cdots x_{n'}}\prod_{i=1}^{n'-1}V_i^{x_{i+1}\cdots x_{n'}}V_{n'}.$$

Therefore, the transcript $\left((U_i, V_i, x_i)_{i=1}^{n'}, \Gamma, \Delta, x_{n'+1}, a', b', u\right)$ of an honest protocol execution is at a statistical distance of at most $2^{-\lambda+2}$ from a tuple

$$\left((U_i, V_i, x_i)_{i=1}^{n'}, f^{2s}, \left(g^{x_{n'+1}\alpha} h^{x_{n'+1}\beta} e^{\alpha\beta} f^t\right)^2 C_{n'+1}^{-x_i^2} f^{-2sx_{n'+1}}, x_{n'+1}, \alpha, \beta, t\right)$$

with, for all $i \in [\![n' + 1]\!]$, $x_i \leftarrow_\$ [\![0; P - 1]\!]$, $s_{u,i}, s_{v,i} \leftarrow_\$ \left[\!\!\left[0; 2n'2^{b_\mathbb{G}+\lambda}\right]\!\!\right]$, $U_i \leftarrow g^{s_{u,i}}$, $V_i \leftarrow g^{s_{v,i}}$, $\alpha, \beta \leftarrow_\$ \left[\!\!\left[0; 2^{\ell+\lambda} P^{n'+1}\right]\!\!\right]$, $s \leftarrow_\$ \left[\!\!\left[0; 2\max(n', 1)2^{b_\mathbb{G}+\lambda}\right]\!\!\right]$, $t \leftarrow_\$ \left[\!\!\left[0; 2\max(n', 1)2^{b_\mathbb{G}+2\lambda} P^{n'+3} + 2^{\ell+\lambda}(P - 1)^{n'+2}\right]\!\!\right]$ and $C_{n'+1}$ computed as above. Note the latter distribution is independent of the witness. Besides, each U_i and V_i can be computed in $O(b_\mathbb{G} + \lambda + \log\log n)$ group operations and the group element $\left(g^{x_{n'+1}\alpha} h^{x_{n'+1}\beta} e^{\alpha\beta} f^t\right)^2 C_{n'+1}^{-x_i^2} f^{-2sx_{n'+1}}$ can be computed in

$$O(\ell + b_\mathbb{G} + \lambda + \log(P)\log(n) + \log\log n) = O(\ell + b_\mathbb{G} + \log(P)\log(n))$$

group operations ($\log P \leq b_\mathbb{G} = \Omega(\lambda)$). A simulated transcript can thus be computed in time $O((\ell + b_\mathbb{G} + \log(P)\log(n)) T_\mathbb{G})$, hence the theorem. □

Lemma 4.3. *Let n be a natural integer and let $a_0, \ldots, a_n, b$ and N be integers, with $N \geq 1$. Assuming that the a_i integers are not all nil modulo N, the number of tuples $(x_0, \ldots, x_n) \in \mathbb{Z}_N^{n+1}$ such that $a_0x_0 + \cdots + a_nx_n + b = 0 \bmod N$ is either 0 or $N^n \gcd(a_0, \ldots, a_n, N)$.*

Proof. The lemma can be proved by induction on n as follows. For $n = 0$, note that the equation $a_0x_0 + b = 0 \bmod N$ has no solution if $\gcd(a_0, N) \nmid b$. If $\gcd(a_0, N) \mid b$, let a_0' be such that $a_0 = \gcd(a_0, N)a_0'$, and define b' and N' similarly. Integers a_0' and N' are then coprime, and let u and v be integers such that $a_0'u + N'v = 1$. Consider now the equation $a_0'x_0 + b' = 0 \bmod N'$. Multiplying it by u shows that $x_0 = -b'u \bmod N'$. It follows that the integers x_0 solutions to the equation $a_0'x_0 + b' = 0 \bmod N'$ are of the form $-b'u + kN'$ for $k \in \mathbb{Z}$. Besides, for two integers k_0 and k_1, $-b'u + k_0N' = -b'u + k_1N' \bmod N$ if and only if $\gcd(a_0, N) \mid k_1 - k_0$. Therefore, if $\gcd(a_0, N) \mid b$, the solutions x_0 to the equation $a_0x_0 + b = 0 \bmod N$ are $-b'u + kN'$ for $0 \leq k \leq \gcd(a_0, N)$.

Now suppose the statement to be true for $n \geq 0$ and consider the equation $a_0x_0 + \cdots + a_{n+1}x_{n+1} + b = 0 \bmod N$ for some integers $a_0, \ldots, a_{n+1}, b$ and $N \geq 1$. For a fixed value $x_{n+1} \in \mathbb{Z}_N$, as in the case $n = 0$, there is no solution if $\gcd(a_0, \ldots, a_n, N) \nmid a_{n+1}x_{n+1} + b$; and if $\gcd(a_0, \ldots, a_n, N) \mid a_{n+1}x_{n+1} + b$, then the induction hypothesis implies that there are $N^n \gcd(a_0, \ldots, a_n, N)$ solutions in $\mathbb{Z}_N$. It now remains to determine the number of values $x_{n+1} \in \mathbb{Z}_N$ such that $a_{n+1}x_{n+1} + b = 0 \bmod \gcd(a_0, \ldots, a_n, N)$. To this end, let u_{n+1} and v_{n+1} be integers such that $a_{n+1}u_{n+1} + \gcd(a_0, \ldots, a_n, N)v_{n+1} = \gcd(a_0, \ldots, a_{n+1}, N)$. As in the case $n = 0$, there is no solution if $\gcd(a_0, \ldots, a_{n+1}, N) \nmid b$, and if $\gcd(a_0, \ldots, a_{n+1}, N) \mid b$, then the solutions to the equation $a_{n+1}x_{n+1} + b = 0 \bmod \gcd(a_0, \ldots, a_n, N)$ are exactly $x_{n+1} = -bu_{n+1} + k_{n+1}d'$ for $0 \leq k_{n+1} < \gcd(a_0, \ldots, a_{n+1}, N)$ and $d' \in \mathbb{Z}$ such that $\gcd(a_0, \ldots, a_n, N) = \gcd(a_0, \ldots, a_{n+1}, N)d'$. The values $x_{n+1} \in \mathbb{Z}_N$ for which the equation $a_0x_0 + \cdots + a_{n+1}x_{n+1} + b = 0 \bmod N$ is satisfied in case

$\gcd(a_0,\ldots,a_{n+1},N) \mid b$ are then exactly $-bu_{n+1}+k_{n+1}d'+k\gcd(a_0,\ldots,a_n,N)$ for $0 \le k_{n+1} < \gcd(a_0,\ldots,a_{n+1},N)$ and $0 \le k < N'$, where $N' \in \mathbb{Z}$ is such that $N = \gcd(a_0,\ldots,a_n,N)N'$. Therefore, the number of solutions to the equation $a_0x_0+\cdots+a_{n+1}x_{n+1}+b = 0 \bmod N$ is either 0 or $N^n\gcd(a_0,\ldots,a_n,N)\gcd(a_0,\ldots,a_{n+1},N)$ $N' = N^{n+1}\gcd(a_0,\ldots,a_{n+1},N)$. The statement of the lemma is then true for all $n \in \mathbb{N}$. □

Lemma 4.4. *Consider the problem (depending on λ) of computing, on input $(\mathbb{G},P) \leftarrow \mathsf{G}\left(1^\lambda\right)$ and $f \leftarrow_\$ \mathbb{G}$ and $(f^{x_i})_{i=0}^n$ (for integers $x_i \leftarrow_\$ \left[\!\left[0;2^{2b_{\mathbb{G}}+\lambda}(n+1)\right]\!\right]$) an element $C \in \mathbb{G}$ and integers $a_0,\ldots,a_n,b,\delta$ such that $1 < |\delta| < P$, δ does not divide b or at least one of the a_i integers, and $C^\delta = f_0^{a_0}\cdots f_n^{a_n}f^b$.*

Under the $(T^{\mathrm{strg}},\varepsilon^{\mathrm{strg}})$-strong-root assumption, the $\left(T^{\mathrm{ord}},\varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G, the probability that any probabilistic algorithm running in time T solves this problem is at most $\left(1/2-2^{-\lambda}-(1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}}+\varepsilon^{\mathrm{strg}}+1-\mu\right)$, if T is such that $(n+1)\max(\log(n+1),1)\log(P)b_{\mathbb{G}}TT_{\mathbb{G}} \le \Omega\left(\min\left(T^{\mathrm{strg}},T^{\mathrm{ord}}\right)\right)$.

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the lemma and assume without loss of generality that $\delta > 0$ (if $\delta < 0$, raise the equality to the power -1). The equality $C^\delta = f_0^{a_0}\cdots f_n^{a_n}f^b$ implies that $C^\delta = f^{\sum_i a_ix_i+b}$. The goal is to show that in case δ does not divide $\sum_i a_ix_i+b$, algorithm $\mathcal{A}$ can be used to violate the assumptions on generator G; and to show that conditioned on the event in which $\mathcal{A}$ solves the problem, the probability that δ divides $\sum_i a_ix_i+b$ is at most $1/2+2^{-\lambda}+(1-\mu)$.

More precisely, if δ does not divide $\sum_i a_ix_i+b$, let $d := \gcd\left(\delta,\sum_i a_ix_i+b\right)$ and $u,v \in \mathbb{Z}$ such that $d = u\delta+v\left(\sum_i a_ix_i+b\right)$. Then, $f^d = (f^uC^v)^\delta$, i.e., $\left((f^uC^v)^{\delta/d}f^{-1}\right)^d = 1_{\mathbb{G}}$. Since $1 \le d < \delta < P$ by assumption, the small-order assumption over G implies that the element $\tilde{g} := (f^uC^v)^{\delta/d}f^{-1}$ is such that $\tilde{g}^2 = 1_{\mathbb{G}}$ with probability at least $\varepsilon^{\mathrm{ord}}$. If $\tilde{g} = 1_{\mathbb{G}}$ and $d > 1$, then $\left((f^uC^v)^{\delta/d},d\right)$ is a solution to the strong-root problem. Otherwise,

- if δ/d is odd, then $\tilde{g}^{\delta/d} = \tilde{g}$ and therefore, $(f^uC^v\tilde{g},\delta/d)$ is a solution to the strong-root problem
- if δ/d is even, then the low-dyadic-valuation assumption on orders implies that $\operatorname{ord}\left((f^uC^v)^{\delta/d}\right)$ is odd, which is impossible if $\operatorname{ord}(f)$ is P-rough (and thus odd) since $\operatorname{ord}(f\tilde{g}) = 2\operatorname{ord}(f)$ in this case.

Consequently, δ does not divide $\sum_i a_ix_i+b$ with probability at most $\varepsilon^{\mathrm{ord}}+\varepsilon^{\mathrm{strg}}+1-\mu$.

Since $|a_i|,|b| \le 2^{O(T)}$, $\sum_i a_ix_i+b$ can be computed in time $O\left((n+1)T(b_{\mathbb{G}}+\log(n+1))\right)$. Then, u and v can be computed in time $O((T+b_{\mathbb{G}}+\log(n+1))\log P)$ with the extended Euclidean algorithm as $\left|\sum_i a_ix_i+b\right| \le n(n+1)2^{O(T)}2^{2b_{\mathbb{G}}+\lambda}+2^{O(T)}$ and $|\delta| \le P$; and u and v are such that $|u|,|v| \le \max\left(|\delta|,\left|\sum_i a_ix_i+b\right|\right)/d$. Besides, computing δ/d can be done in time $O\left(\log^2 P\right)$ and then $f^uC^v\tilde{g}$ in $O\left(\max\left(T+b_{\mathbb{G}}+\log(n+1),\log P\right)\right) = O\left(T+b_{\mathbb{G}}+\log(n+1)\right)$ group operations since $P \leq 2^{b_{\mathbb{G}}}$. The solution to the strong-root problem can thus be computed in time

$$O\left((n+1)(b_{\mathbb{G}} + \log(n+1))T + (T + b_{\mathbb{G}} + \log(n+1))\log(P)T_{\mathbb{G}}\right),$$

after the bases $f_0, \ldots, f_n$ have been computed in $O((n+1)\max(\log(n+1), 1)b_{\mathbb{G}})$ group operations.

It remains to show that δ divides $\sum_i a_i x_i + b$ with probability at most $1/2 + 2^{-\lambda} + 1 - \mu$ conditioned on the event in which $\mathscr{A}$ solves the problem. To do so, consider the event in which it occurs. Let p and j respectively be a prime and a positive integer such that p^j divides δ and p^j does not divide b or at least one of the a_i integers. Such p and j necessarily exist for an assumption of the lemma is that δ does not divide b or at least one of the a_i integers. Note that p^j cannot divide all the a_i integers as it would otherwise divide b as well, since it divides $\sum_i a_i x_i + b$. Moreover, if μ-assumption that there are many rough-order elements in the groups generated by G holds, p does not divide $\operatorname{ord}(f)$. Therefore, if the μ-assumption holds, p^j does not divide $a_i \operatorname{ord}(f)$ for some $i \in [\![0; n]\!]$.

For $i \in [\![0; n]\!]$, let $0 \leq \rho_i < \operatorname{ord}(f)$ be the unique integer such that $x_i = \operatorname{ord}(f) \lfloor x_i / \operatorname{ord}(f) \rfloor + \rho_i$, and note that $f^{x_i} = f^{\rho_i}$. Then, $\sum_i a_i x_i + b = \sum_i a_i \operatorname{ord}(f) \lfloor x_i / \operatorname{ord}(f) \rfloor + \sum_i a_i \rho_i + b = 0 \bmod p^j$ and $a_i \operatorname{ord}(f) \neq 0 \bmod p^j$ for some $i \in [\![0; n]\!]$. Lemma 4.3 shows that the equation $\sum_i A_i X_i + B = 0 \bmod p^j$ with $A_i \coloneqq a_i \operatorname{ord}(f)$ and $B \coloneqq \sum_i a_i \rho_i + b$ has at most $p^{jn} \gcd\left(a_0 \operatorname{ord}(f), \ldots, a_n \operatorname{ord}(f), p^j\right)$ solutions, and $\gcd\left(a_0 \operatorname{ord}(f), \ldots, a_n \operatorname{ord}(f), p^j\right)$ is at most p^{j-1} since $a_i \operatorname{ord}(f) \neq 0 \bmod p^j$ for some $i \in [\![0; n]\!]$. However, the variables $X_i \coloneqq \lfloor x_i / \operatorname{ord}(f) \rfloor$ are identically distributed and independent of the values returned by $\mathscr{A}\left(\mathbb{G}, P, f, f^{\rho_0}, \ldots, f^{\rho_n}\right)$; and their distribution is at a statistical distance of at most $\operatorname{ord}(f) 2^{-2b_{\mathbb{G}}-\lambda}(n+1)^{-1} \leq 2^{-b_{\mathbb{G}}-\lambda}(n+1)^{-1}$ from the uniform distribution over $\left[\!\!\left[0; \left\lfloor (n+1)2^{2b_{\mathbb{G}}+\lambda} / \operatorname{ord}(f) \right\rfloor \right]\!\!\right] \supseteq \left[\!\!\left[0; (n+1)2^{b_{\mathbb{G}}+\lambda}\right]\!\!\right]$. Besides, if a variable X is uniformly distributed over the set $\left[\!\!\left[0; (n+1)2^{b_{\mathbb{G}}+\lambda}\right]\!\!\right]$, then the distribution of $X \bmod p^j$ is at a statistical of at most $p^j 2^{-b_{\mathbb{G}}-\lambda}(n+1)^{-1} \leq (P-1)2^{-b_{\mathbb{G}}-\lambda}(n+1)^{-1}$ from the uniform distribution over $\mathbb{Z}_{p^j}$. The distribution of the random vector $\left(X_0 \bmod p^j, \ \ldots, X_n \bmod p^j\right)$ is then at a statistical of at most $P 2^{-b_{\mathbb{G}}-\lambda} \leq 2^{-\lambda}$ from the uniform distribution over $\mathbb{Z}_{p^j}^{n+1}$. Consequently, the equation $\sum_i a_i x_i + b = 0 \bmod p^j$ can then be satisfied with probability at most $2^{-\lambda} + p^{j(n+1)-1} / \left(p^j\right)^{n+1} \leq 1/2 + 2^{-\lambda}$ and thus, δ divides $\sum_i a_i x_i + b$ with probability at most $1/2 + 2^{-\lambda} + 1 - \mu$.

In summary, denoting by ε the probability that $\mathscr{A}$ solves the problem of the statement of the lemma, $\varepsilon \leq \varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu + \left(1/2 + 2^{-\lambda} + 1 - \mu\right)\varepsilon$, which is equivalent to $\varepsilon \leq \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)$. □

Lemma 4.5 (Discrete-Logarithm Relations). *Let n be a non-negative integer. Consider the problem (depending on λ) of computing, on the input of $(\mathbb{G}, P) \leftarrow \mathsf{G}\left(1^\lambda\right)$ and of group elements $f \leftarrow_\$ \mathbb{G}$ and $\left(f^{x_i}\right)_{i=0}^n$ (for $x_i \leftarrow_\$ \left[\!\!\left[0; 2^{2b_{\mathbb{G}}+\lambda}(n+1)\right]\!\!\right]$), integers $a_0, \ldots, a_n, b$ such that $f_0^{a_0} \cdots f_n^{a_n} f^b = 1_{\mathbb{G}}$ although at least one of $a_0, \ldots, a_n, b$ is non-zero. Under the $\left(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}}\right)$-strong-root assumption, the $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and* *the μ-assumption over G, the probability that any probabilistic algorithm running in time at most T solves this problem is at most*

$$\varepsilon^{\mathrm{strg}} + \max\left(2^{-b_{\mathbb{G}}-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} \ +1 - \mu\right)\right)$$

if T is such that $(n+1)\max(\log(n+1), 1)\log(P) b_{\mathbb{G}} T T_{\mathbb{G}} \leq \Omega\left(\min\left(T^{\mathrm{strg}}, T^{\mathrm{ord}}\right)\right)$.

Proof. Let $\mathcal{A}$ be an algorithm as in the statement of the lemma and denote the probability that it solves the problem by ε. If $a_0 = \cdots = a_n = 0$, then $b \neq 0$ by assumption and Lemma 3.4 shows that since $f^b = 1_{\mathbb{G}}$, there exists an algorithm that solves the strong-root problem in time at most $T + O(\log b)$ with probability at least ε, and since $b = 2^{O(T)}$, $\varepsilon \leq \varepsilon^{\mathrm{strg}}$. Now turn to the case in which $a_i \neq 0$ for some $i \in [\![0;n]\!]$. If $n = 0$, then $f^{a_0 x_0 + b} = 1_{\mathbb{G}}$ by assumption. Writing x_0 as $x_0 = \operatorname{ord}(f) \lfloor x_0 / \operatorname{ord}(f) \rfloor + \rho_0$ for $0 \leq \rho_0 < \operatorname{ord}(f)$, the random variable $X_0 := \lfloor x_0 / \operatorname{ord}(f) \rfloor$ is independent of the values returned by $\mathcal{A}(\mathbb{G}, P, f, f^{\rho_0})$, and is at a statistical distance of at most $\operatorname{ord}(f) 2^{-2b_{\mathbb{G}}-\lambda} \leq 2^{-b_{\mathbb{G}}-\lambda}$ from the uniform distribution over $\left[\!\!\left[0; \left\lfloor 2^{2b_{\mathbb{G}}+\lambda} / \operatorname{ord}(f) \right\rfloor\right]\!\!\right] \supseteq \left[\!\!\left[0; 2^{b_{\mathbb{G}}+\lambda}\right]\!\!\right]$. However, for $A_0 := a_0 \operatorname{ord}(f)$ and $B := a_0 \rho_0 + b$, the equation $A_0 X_0 + B = 0$ in $\mathbb{Z}$ has no solution if $A_0 \nmid B$ and exactly one otherwise. Therefore, the probability that $a_0 x_0 + b = 0$ in $\mathbb{Z}$ is at most $2^{-b_{\mathbb{G}}-\lambda+1}$, and there exists an algorithm that solves the strong-root problem in time at most $O(T)$ with probability at least $\varepsilon - 2^{-b-\mathbb{G}-\lambda+1}$, so $\varepsilon \leq \varepsilon^{\mathrm{strg}} + 2^{-b_{\mathbb{G}}-\lambda+1}$.

If $n > 0$, it suffices to prove that the probability that $f_0^{a_0} \cdots f_n^{a_n} f^b = 1_{\mathbb{G}}$ and $\sum_i a_i x_i + b = 0$ is at most $\left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)$. Then, in case $f^{\sum_i a_i x_i + b} = 1_{\mathbb{G}}$ and $\sum_i a_i x_i + b \neq 0$, Lemma 3.4 shows that this probability is at most $\varepsilon^{\mathrm{strg}}$. This then would imply that

$$\varepsilon \leq \varepsilon^{\mathrm{strg}} + \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right).$$

Suppose that $\sum_i a_i x_i + b = 0$ (which implies that $f_0^{a_0} \cdots f_n^{a_n} f^b = 1_{\mathbb{G}}$). Let $d := \gcd(a_0, \ldots, a_n)$ and note that d necessarily divides b. Besides, $\sum_i a_i x_i + b = 0$ if and only if $\sum_i (a_i/d) x_i + (b/d) = 0$ and therefore, $f_0^{a_0/d} \cdots f_n^{a_n/d} f^{b/d} = 1_{\mathbb{G}}$ with $\gcd(a_0/d, \ldots, a_n/d) = 1$. However, $1_{\mathbb{G}}^2 = 1_{\mathbb{G}} = f_0^{a_0/d} \cdots f_n^{a_n/d} f^{b/d}$ although the integers a_i/d cannot all be even as they are coprime. Lemma 4.4 then implies that $\sum_i a_i x_i + b = 0$ with probability at most $\left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} \ +1 - \mu\right)$. □

Theorem 4.6 (Extractability of Π'). *Under the $(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}})$-strong-root assumption, the $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G, protocol Π' (with honest CRS generation) is $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, \varepsilon^{\mathrm{ext}}, \Sigma')$-extractable for any $T_{\mathcal{A}}$ and T_{Prove^*} such that* $T_{\mathcal{A}} + n \log(n+1) \log(P) b_{\mathbb{G}} T_{\mathsf{Prove}^*} T_{\mathbb{G}} \leq \Omega\left(\min\left(T^{\mathrm{strg}}, T^{\mathrm{ord}}\right)\right)$, *with*

$$T_{\mathcal{E}} = O\left(n b_{\mathbb{G}} T_{\mathbb{G}} + n^{\log 5 + \log \alpha} \log(n+1) \log(P) T_{\mathsf{Prove}^*} / \varepsilon\right)$$

for any $\alpha > (1-5/P)^{-2}$ *assuming that* $\varepsilon_{\mathcal{A},\mathsf{Prove}^*} \geq 5n^{\log \alpha}/((\alpha-1)P)$*, and with*

$$\varepsilon^{\mathrm{ext}} = \varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + \max\left(2^{-b-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)\right)$$
$$+ \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right).$$

Proof. To prove that the system satisfies witness-extended emulation, it suffices to show that at the $(n'+1)$th step of the protocol, a witness for commitment $C_{n'+1}$ can be extracted by rewinding the prover, and then that for i from $n'+1$ down to 2, given a witness for C_i, a witness for C_{i-1} can again be extracted by rewinding the prover sufficiently many times. One a witness for the initial commitment C_1 is extracted, it is then possible to generate a transcript with the same distribution as in the interaction with the honest verifier since the protocol is public coin.

Henceforward, and until the end of this proof, the underscript i indicating the current step of the protocol will be omitted as the specific step should always be clear from the context. The underscripts in what follows rather indicate several challenges at the *same* protocol step. It might be convenient for the reader to think of these latter as underscripts j; for instance, at the ith step of the protocol, x_1, x_2 and x_3 are actually challenges $x_{i,1}$, $x_{i,2}$ and $x_{i,3}$, i.e., $x_{i,j}$ for $j = 1, 2, 3$.

Now consider the case $i = n'+1$ (recall that $n' := \lceil \log n \rceil$). Given five transcripts $\left(\Gamma, \Delta, x_j, a'_j, b'_j, u_j\right)_{j=1}^5$ such that $\left(g^{x_j a'_j} h^{x_j b'_j} e^{a'_j b'_j} f^{u_j}\right)^4 = \left(C^{x_j^2}\Gamma^{x_j}\Delta\right)^2$ for all $j \in [\![5]\!]$, the goal is to extract a representation of C in the bases g, h, e and f. To do so, consider the linear system

$$\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

with unknowns v_1, v_2 and v_3. Denote by $\mathbf{X}$ the matrix $\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix}$. It is a Vandermonde matrix, and its determinant is thus $(x_3-x_2)(x_3-x_1)(x_2-x_1)$. Even if $\det \mathbf{X} \neq 0$, this system may not have a solution in $\mathbb{Z}$, but it does in $\mathbb{Q}$. Moreover, Cramer's rule implies that

$$v_1 \det \mathbf{X} = \det \begin{bmatrix} 0 & 1 & 1 \\ 0 & x_2 & x_3 \\ 1 & x_2^2 & x_3^2 \end{bmatrix}, \quad v_2 \det \mathbf{X} = \det \begin{bmatrix} 1 & 0 & 1 \\ x_1 & 0 & x_3 \\ x_1^2 & 1 & x_3^2 \end{bmatrix} \text{ and } v_3 \det \mathbf{X} = \det \begin{bmatrix} 1 & 1 & 0 \\ x_1 & x_2 & 0 \\ x_1^2 & x_2^2 & 1 \end{bmatrix}.$$

It follows that

$$v_1 \overbrace{(x_3-x_1)(x_2-x_1)}^{\delta_1}(x_3-x_2) = \overbrace{1}^{\gamma_1}(x_3-x_2)$$
$$v_2 \overbrace{(x_3-x_2)(x_1-x_2)}^{\delta_2}(x_3-x_1) = \overbrace{1}^{\gamma_2}(x_3-x_1)$$
$$v_3 \overbrace{(x_2-x_3)(x_1-x_3)}^{\delta_3}(x_2-x_1) = \overbrace{1}^{\gamma_3}(x_2-x_1)$$

and that

$$\det \mathbf{X} \sum_j \delta^j \gamma_j x_j^2 = \delta \det \mathbf{X}, \ \sum_j \delta^j \gamma_j x_j = 0 \ \text{ and } \ \sum_j \delta^j \gamma_j = 0$$

for $\delta \coloneqq \delta_1 \delta_2 \delta_3$ and $\delta^j \coloneqq \delta / \delta_j$ for $j \in [\![3]\!]$; and δ, δ^j and γ_j are in $\mathbb{Z}$ for all $j \in [\![3]\!]$. Besides, note that $|\delta| \leq P^3 \leq P'$. In other words, the linear system

$$\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \delta \end{bmatrix}$$

has a (unique) solution in $\mathbb{Z}$ if $\det \mathbf{X} \neq 0$, and it is $\begin{bmatrix} \delta^1 \gamma_1 \\ \delta^2 \gamma_2 \\ \delta^3 \gamma_3 \end{bmatrix}$. Therefore, assuming x_1, x_2 and x_3 to be pairwise distinct, one can extract, via linear combinations of the responses, integers a_C, b_C, z_C and r_C such that $C^{2\delta} = \left(g^{a_C} h^{b_C} e^{z_C} f^{r_C} \right)^4$, and $\delta \neq 0$; and the bit length of the linear coefficients is of order $O(\log P)$. However, g can be expressed in terms of $g_1, \ldots, g_n$ and $x_1, \ldots, x_n$, and the exponent of g_n in the expression of g is 1. Similarly, h can be expressed in terms of $h_1, \ldots, h_n$ and $x_1, \ldots, x_n$, and the exponent of h_1 in the expression of h is 1. Lemma 4.4 then implies that if $|\delta| > 1$, 2δ divides $4a_C$, $4b_C$, $4z_C$ and $4r_C$ with probability at least $1 - \left(1/2 - 2^{-\lambda} - (1 - \mu)\right)^{-1} \left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)$. The small-order assumption then implies that $C = \left(g^{a_C/\delta} h^{b_C/\delta} e^{z_C/\delta} f^{r_C/\delta} \right)^2 \tilde{g}_C$ for some group element $\tilde{g}_C$ such that $\tilde{g}_C^2 = 1_{\mathbb{G}}$, i.e., up to a relabeling of the integers $2a_C/\delta$, $2b_C/\delta$, $2z_C/\delta$ and $2r_C/\delta$, there exist a group element $\tilde{g}_C$ and integers a_C, b_C, z_C and r_C such that $C = \left(g^{a_C} h^{b_C} e^{z_C} f^{r_C} \right)^2 \tilde{g}_C$ and $\tilde{g}_C^2 = 1_{\mathbb{G}}$.

Likewise, by considering the linear systems $\mathbf{X} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ and $\mathbf{X} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, with probability at least the same probability as above, there exist group elements $\tilde{g}_\Gamma$ and $\tilde{g}_\Delta$ and integers a_Γ, a_Δ, b_Γ, b_Δ, z_Γ, z_Δ, r_Γ and r_Δ such that $\Gamma = \left(g^{a_\Gamma} h^{b_\Gamma} e^{z_\Gamma} f^{r_\Gamma} \right)^2 \tilde{g}_\Gamma$, $\Delta = \left(g^{a_\Delta} h^{b_\Delta} e^{z_\Delta} f^{r_\Delta} \right)^2 \tilde{g}_\Delta$ and $\tilde{g}_\Gamma^2 = \tilde{g}_\Delta^2 = 1_{\mathbb{G}}$.

Consequently, for any $(x, a', b', u) \in \left\{ \left(x_j, a'_j, b'_j, u_j \right)_{j=1}^5 \right\}$,

$$\begin{aligned} \left(g^{xa'} h^{xb'} e^{a'b'} f^u \right)^4 = &\left(g^{a_C} h^{b_C} e^{z_C} f^{r_C} \right)^{4x^2} \left(g^{a_\Gamma} h^{b_\Gamma} e^{z_\Gamma} f^{r_\Gamma} \right)^{4x} \\ &\left(g^{a_\Delta} h^{b_\Delta} e^{z_\Delta} f^{r_\Delta} \right)^4, \end{aligned}$$

Furthermore, Lemma 4.5 entails that

$$\begin{aligned} a'x &= a_C x^2 + a_\Gamma x + a_\Delta \\ b'x &= b_C x^2 + b_\Gamma x + a_\Delta \\ a'b' &= z_C x^2 + z_\Gamma x + z_\Delta \\ u &= r_C x^2 + r_\Gamma x + r_\Delta \end{aligned}$$

unless one can find a non-trivial discrete-logarithm relation in $\langle f \rangle$. That is because the exponent of g_n is 1 in the expression of g in terms of the initial group elements $g_1, \ldots, g_n$ and of the challenges $x_1, \ldots, x_{n'}$, and since the exponent of h_1 in the expression of h is 1 in terms of $h_1, \ldots, h_n$ and of $x_1, \ldots, x_{n'}$. Multiplying the third equality by x^2 then implies that

$$\left(a_C x^2 + a_\Gamma x + a_\Delta\right)\left(b_C x^2 + b_\Gamma x + a_\Delta\right) - z_C x^4 + z_\Gamma x^3 + z_\Delta x^2 = 0,$$

i.e., the above polynomial in $\mathbb{Z}[x]$ is of degree 4 and has at least 5 integer roots if the challenges $x_1, \ldots, x_5$ are pairwise distinct. It is thus the zero polynomial, $a_C b_C = z_C$ and (a_C, b_C, r_C) is a valid culpable witness for C.

It now remains to show that for i from n' down to 1, given a witness for the commitment at the $i + 1$th step, a witness for the commitment at the ith step can be extracted. To this end, consider five transcripts $\left(U, V, x_j, \mathbf{a}'_j, \mathbf{b}'_j, t'_j\right)_{i=1}^5$ such that

$$\left(\left(\mathbf{g}_1{}^{x_j} \circ \mathbf{g}_2\right)^{\mathbf{a}'_j} \left(\mathbf{h}_1 \circ \mathbf{h}_2{}^{x_j}\right)^{\mathbf{b}'_j} e^{\left\langle \mathbf{a}'_j, \mathbf{b}'_j \right\rangle} f^{t_j}\right)^4 = \left(U^{x_j^2} C^{x_j} V\right)^2$$

for all $j \in [\![5]\!]$. The objective is to find an expression for C in the bases $\mathbf{g}_1$, $\mathbf{g}_2$, $\mathbf{h}_1$, $\mathbf{h}_2$, e and f.

As in the case $i = n' + 1$, by considering the linear systems $\mathbf{X}\begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, $\mathbf{X}\begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ and $\mathbf{X}\begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, one can extract integers $\delta_U, \delta_C, \delta_V$, z_U, z_C, z_V, r_U, r_C, r_V, and integer vectors $\mathbf{a}_U, \mathbf{a}_C, \mathbf{a}_V$, $\mathbf{b}_U, \mathbf{b}_C, \mathbf{b}_V \in \mathbb{Z}^{\lceil n 2^{-i+2} \rceil}$ such that

$$\begin{aligned}
U^{2\delta_U} &= \left(\mathbf{g}^{\mathbf{a}_U} \mathbf{h}^{\mathbf{b}_U} e^{z_U} f^{r_U}\right)^4 \\
C^{2\delta_C} &= \left(\mathbf{g}^{\mathbf{a}_C} \mathbf{h}^{\mathbf{b}_C} e^{z_C} f^{r_C}\right)^4 \\
V^{2\delta_V} &= \left(\mathbf{g}^{\mathbf{a}_V} \mathbf{h}^{\mathbf{b}_V} e^{z_V} f^{r_V}\right)^4.
\end{aligned}$$

Moreover, if the challenges are pairwise distinct, $\delta_U, \delta_C, \delta_V \neq 0$. Lemma 4.4 and the small-order assumption imply that one can actually extract integers z_U, z_C, z_V, r_U, r_C, r_V, and integer vectors $\mathbf{a}_U, \mathbf{a}_C, \mathbf{a}_V$, $\mathbf{b}_U, \mathbf{b}_C, \mathbf{b}_V \in \mathbb{Z}^{\lceil n 2^{-i+2} \rceil}$ such that

$$\begin{aligned}
U &= \left(\mathbf{g}^{\mathbf{a}_U} \mathbf{h}^{\mathbf{b}_U} e^{z_U} f^{r_U}\right)^2 \tilde{g}_U \\
C &= \left(\mathbf{g}^{\mathbf{a}_C} \mathbf{h}^{\mathbf{b}_C} e^{z_C} f^{r_C}\right)^2 \tilde{g}_C \\
V &= \left(\mathbf{g}^{\mathbf{a}_V} \mathbf{h}^{\mathbf{b}_V} e^{z_V} f^{r_V}\right)^2 \tilde{g}_V
\end{aligned}$$

for some $\tilde{g}_U, \tilde{g}_C, \tilde{g}_V \in \mathbb{G}$ that satisfy $\tilde{g}_U^2 = \tilde{g}_C^2 = \tilde{g}_V^2 = 1_{\mathbb{G}}$.

Therefore, for any $(x, \mathbf{a}', \mathbf{b}', t') \in \left\{\left(x_j, \mathbf{a}'_j, \mathbf{b}'_j, t'_j\right)_{i=1}^5\right\}$,

$$\begin{aligned}
1_{\mathbb{G}} = {} & \mathbf{g}_1{}^{4\left(x\mathbf{a}' - \mathbf{a}_{U,1} x^2 - \mathbf{a}_{C,1} x - \mathbf{a}_{V,1}\right)} \mathbf{g}_2{}^{4\left(\mathbf{a}' - \mathbf{a}_{U,2} x^2 - \mathbf{a}_{C,2} x - \mathbf{a}_{V,2}\right)} \mathbf{h}_1{}^{4\left(\mathbf{b}' - \mathbf{b}_{U,1} x^2 - \mathbf{b}_{C,1} x - \mathbf{b}_{V,1}\right)} \\
& \mathbf{h}_2{}^{4\left(x\mathbf{b}' - \mathbf{b}_{U,2} x^2 - \mathbf{b}_{C,2} x - \mathbf{b}_{V,2}\right)} e^{4\left(\langle \mathbf{a}', \mathbf{b}' \rangle - z_U x^2 - z_C x - z_V\right)} f^{4\left(t' - r_U x^2 - r_C x - r_V\right)}.
\end{aligned}$$

Lemma 4.5 then implies that

$$
\begin{align}
x\mathbf{a}' &= \mathbf{a}_{U,1}x^2 + \mathbf{a}_{C,1}x + \mathbf{a}_{V,1} \tag{1}\\
\mathbf{a}' &= \mathbf{a}_{U,2}x^2 + \mathbf{a}_{C,2}x + \mathbf{a}_{V,2} \tag{2}\\
\mathbf{b}' &= \mathbf{b}_{U,1}x^2 + \mathbf{b}_{C,1}x + \mathbf{b}_{V,1} \tag{3}\\
x\mathbf{b}' &= \mathbf{b}_{U,2}x^2 + \mathbf{b}_{C,2}x + \mathbf{b}_{V,2} \tag{4}\\
\langle \mathbf{a}', \mathbf{b}' \rangle &= z_U x^2 + z_C x + z_V \tag{5}\\
t' &= r_U x^2 + r_C x + r_V \tag{6}
\end{align}
$$

since the expressions of the components of $\mathbf{g}_1, \mathbf{g}_2, \mathbf{h}_1, \mathbf{h}_2$ in terms of the challenges $x_1, \ldots, x_{i-1}$ (if $i \geq 2$) and of the initial bases $g_1, \ldots, g_n, h_1, \ldots, h_n$ are such that for each component, there exists an initial basis with 1 as an exponent. Indeed, if $i = 1$, then the statement is true by definition. If $i \geq 2$, denote by $\mathbf{g}(i)$ and $\mathbf{h}(i)$ the vectors at step i. Then, $\mathbf{g}(i) = \mathbf{g}(i-1)_1^{x_{i-1}} \circ \mathbf{g}(i)_2$ and $\mathbf{h}(i) = \mathbf{h}(i-1)_1 \circ \mathbf{h}(i-1)_2^{x_{i-1}}$, which shows that if the statement is true for $i-1$, then it is true for i and the statement is then true for all $i \in [\![n'+1]\!]$.

Equations 1 and 2 imply that

$$\mathbf{a}_{U,2}x^3 + \left(\mathbf{a}_{C,2} - \mathbf{a}_{U,1}\right) x^2 + \left(\mathbf{a}_{V,2} - \mathbf{a}_{C,1}\right) x - \mathbf{a}_{V,1} = 0. \tag{7}$$

Similarly, Equations 3 and 4 entail that

$$\mathbf{b}_{U,1}x^3 + \left(\mathbf{b}_{C,1} - \mathbf{b}_{U,2}\right) x^2 + \left(\mathbf{b}_{V,1} - \mathbf{b}_{C,2}\right) x - \mathbf{b}_{V,2} = 0. \tag{8}$$

Besides, from Equations 2, 3 and 5,

$$\left\langle \mathbf{a}_{U,2}x^2 + \mathbf{a}_{C,2}x + \mathbf{a}_{V,2}, \mathbf{b}_{U,1}x^2 + \mathbf{b}_{C,1}x + \mathbf{b}_{V,1} \right\rangle - z_U x^2 + z_C x + z_V = 0, \tag{9}$$

and the coefficient of x in this polynomial is

$$\langle \mathbf{a}_{C,2}, \mathbf{b}_{V,1} \rangle + \langle \mathbf{a}_{V,2}, \mathbf{b}_{C,1} \rangle - z_C.$$

If the 5 challenges are pairwise distinct, the polynomials in Equations 7, 8 and 9 are necessarily nil as they are of degree at most 4. Therefore, $\mathbf{b}_{V,1} = \mathbf{b}_{C,2}$, $\mathbf{a}_{V,2} = \mathbf{a}_{C,1}$ and $z_C = \langle \mathbf{a}_C, \mathbf{b}_C \rangle$. That is, $(\mathbf{a}_C, \mathbf{b}_C, r_C)$ is a valid witness for C.

Given a prover $(\mathcal{A}, \mathsf{Prove}^*)$ with success probability at least ε, define then $\mathcal{E}$ as an algorithm which first generates bases $\mathbf{g}, \mathbf{h}, e$ as in the real protocol (which requires $O(nb_{\mathbb{G}})$ group operations). It continues by running the challenge-tree generator of del Pino, Seiler and Lyubashevsky's forking lemma [16] (corrected in Section 4.2) with blackbox access to Prove^*. Since the extractor needs 5 transcripts at each recursion step, their challenge-tree generator runs in time $O\left(n^{\log 5 + \log \alpha} \log(n+1) T_{\mathsf{Prove}^*}/\varepsilon\right)$ for any $\alpha > (1 - 5/P)^{-2}$, assuming that the prover succeeds with probability at least $5n^{\log \alpha}/((\alpha - 1)P)$. Then, algorithm $\mathcal{E}$ can extract a valid witness in time $O(n \log(n+1) \log(P) T_{\mathsf{Prove}^*})$ unless the extraction fails at one of the protocol steps, since a witness at each recursion step can be computed via linear combinations with coefficients of $O(\log P)$ bits of integer vectors of length n derived from integers returned by Prove^* (so at most $2^{O(T_{\mathsf{Prove}^*})}$ in absolute value). Extraction fails with probability at most

$$\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + \max\left(2^{-b-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)\right) + \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right),$$

hence the theorem. □

Corollary 4.7 (Extractability of Π). *Under the $(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}})$-strong-root assumption, the $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G, protocol Π is $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, q_{\mathcal{H}}, \varepsilon^{\mathrm{ext}}, \Sigma)$-extractable for any $T_{\mathcal{A}}$ and T_{Prove^*} such that $T_{\mathcal{A}} + n\log(n+1)\log(P)b_{\mathbb{G}}T_{\mathsf{Prove}^*}T_{\mathbb{G}} \leq \Omega\left(\min\left(T^{\mathrm{strg}}, T^{\mathrm{ord}}\right)\right)$, with $T_{\mathcal{E}} = 3T_{\tilde{\Pi}^{\mathcal{H}}.\mathsf{Sim}} + 2T_{\Pi'.\mathcal{E}} + O((b + \log n + \ell)T_{\mathbb{G}})$ and*

$$\varepsilon^{\mathrm{ext}} = \varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + \varepsilon^{\mathrm{zk}}_{\tilde{\Pi}} + \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right) + \max\left(2^{-b-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)\right),$$

if $FS.\tilde{\Pi}^{\mathcal{H}}$ is $\left(T_{\tilde{\Pi}^{\mathcal{H}}.\mathsf{Sim}}, q_{\mathcal{H}}, \varepsilon^{\mathrm{zk}}_{\tilde{\Pi}^{\mathcal{H}}}\right)$-statistically honest-verifier zero-knowledge.

Proof. Given integers α, α', r, r' and integer vectors $\mathbf{a}, \mathbf{a}', \mathbf{b}, \mathbf{b}'$ such that $\left(Cf^{2\alpha z}\right)^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} f^{\alpha\langle\mathbf{a},\mathbf{b}\rangle} f^r\right)^4$ and $\left(Cf^{2\alpha' z}\right)^2 = \left(\mathbf{g}^{\mathbf{a}'}\mathbf{h}^{\mathbf{b}'} f^{\alpha'\langle\mathbf{a}',\mathbf{b}'\rangle} f^{r'}\right)^4$, it follows that

$$1_{\mathbb{G}} = \left(\mathbf{g}^{\mathbf{a}-\mathbf{a}'}\mathbf{h}^{\mathbf{b}-\mathbf{b}'} f^{\alpha(\langle\mathbf{a},\mathbf{b}\rangle - z)} f^{\alpha'(\langle\mathbf{a}',\mathbf{b}'\rangle - z)} f^{r-r'}\right)^4.$$

Lemma 4.5 shows that if $\alpha, \alpha' \leftarrow_{\$} \left[\!\left[0; 2^{b+2\lambda}\right]\!\right]$, then $\mathbf{a} = \mathbf{a}'$, $\mathbf{b} = \mathbf{b}'$ and $z = \langle\mathbf{a}, \mathbf{b}\rangle$ unless one can find a non-trivial discrete-logarithm relation in $\langle f\rangle$.

with probability at least

$$1 - \varepsilon^{\mathrm{strg}} - \max\left(2^{-b-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)\right).$$

Consider then an algorithm $\Pi.\mathcal{E}$ that generates bases as in the real protocol but simulates π_{crs} and π with $FS.\tilde{\Pi}^{\mathcal{H}}.\mathsf{Sim}$ (the time to generate the bases is already included in $T_{\Pi'.\mathcal{E}}$). It generates $\alpha \leftarrow_{\$} \left[\!\left[0; 2^{b+2\lambda}\right]\!\right]$, computes $Cf^{2\alpha z}$ and runs $\Pi'.\mathcal{E}$ with Prove^* as subroutine from the computation step after this latter has verified π, on the inputs of protocol Π'. It then obtains an integer r and integer vectors $\mathbf{a}$ and $\mathbf{b}$ such that $\left(Cf^{2\alpha z}\right)^2 = \left(\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} f^{\alpha\langle\mathbf{a},\mathbf{b}\rangle} f^r\right)^4$. It generates anew $\alpha' \leftarrow_{\$} \left[\!\left[0; 2^{b+2\lambda}\right]\!\right]$, computes $Cf^{2\alpha' z}$, simulates π and runs $\Pi'.\mathcal{E}$ as before a second time. Since $\Pi'.\mathcal{E}$ cannot extract a valid witness for Σ' with probability at most

$$\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + \max\left(2^{-b-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)\right) + \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right),$$

the analysis above then shows that $\Pi.\mathcal{E}$ cannot extract a valid witness for Σ with probability at most

$$\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + \varepsilon^{\mathrm{zk}}_{\tilde{\Pi}} + \max\left(2^{-b-\lambda+1}, \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right)\right) + \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right).$$

Note that the bounds on the failure probability of $\Pi'.\mathcal{E}$ already accounted for the probability of finding non-trivial discrete-logarithm relations in $\langle f \rangle$.

As $z = \langle \mathbf{a}, \mathbf{b} \rangle$ and the components of $\mathbf{a}$ and $\mathbf{b}$ are of absolute value less than 2^{ℓ}, $|z| \leq n2^{2\ell}$ and $Cf^{2\alpha z}$ and $Cf^{2\alpha' z}$ can both be computed in $O(b_{\mathbb{G}} + \ell + \log n)$ group operations. Therefore, $\Pi.\mathcal{E}$ runs in time $3T_{\tilde{\Pi}^{\mathcal{H}}.\mathsf{Sim}} + 2T_{\Pi'.\mathcal{E}} + O((b + \log n + \ell)T_{\mathbb{G}})$ and the theorem follows. □

Challenge-Tree Generators. For any prover $(\mathcal{A}, \mathsf{Prove}^*)$ with success probability at least $\varepsilon > 0$, the challenge-tree generator of del Pino, Seiler and Lyubashevsky's forking lemma [16] is claimed to run in time $O\left(n^{\log m + \log \alpha} \log(n)\right.$ $T_{\mathsf{Prove}^*}/\varepsilon)$ for any $\alpha > (1 - 5/P)^{-2}$, with m denoting the arity of the tree that must be generated. However, we explain that their analysis does not apply to their algorithm, and show how to modify their algorithm to obtain the claimed bound.

More precisely, the task of their algorithm is to generate a rooted tree of height $n' = \lceil \log n \rceil$ (the number of recursion steps) in which each node at height $1 \leq i \leq n' - 1$ has m_i children ($m = \max m_i$), and to label each node with a challenge so that no two siblings in the tree have the same label. Moreover, the executions of Prove^* with the challenges on the paths from the root to each leaf must be accepting. These conditions must all be met for the extractor of the Bulletproofs to be bale extract a witness at each recursion step.

To generate such challenge trees, Bootle et al. [8] had proposed a recursive tree-finder algorithm that, given fixed challenges $x_1, \ldots, x_{i-1}$, generates a challenge x_i uniformly at random at each recursion step $i > 1$ and makes a recursive call with $x_1, \ldots, x_i$. A challenge is then appended to the tree if the recursive call returns a non-empty tree. At the last recursion step (i.e., at each leaf of the tree), the algorithm runs the protocol with the generated challenge and returns the challenge (as single node) only if the execution is successful. At each recursion step $i < n'$, the algorithm repeats the procedure of generating fresh challenges and making recursive calls as many times as necessary to obtain m_i challenges that eventually lead to a success.

The main issue with their algorithm as pointed out by del Pino, Seiler and Lyubashevsky is that for challenges $x_1, \ldots, x_{i-1}$, the prover may only have negligible success probability if the first $i-1$ challenges are fixed to $x_1, \ldots, x_{i-1}$, even though the overall success probability of the prover might be non-negligible if all challenges are generated uniformly at random. As a consequence, in case such $x_1, \ldots, x_{i-1}$ is chosen, the tree-finder algorithm may run for an arbitrarily long time.

To resolve this issue, del Pino, Seiler and Lyubashevsky's idea was then to estimate the probability that a certain x_i leads to a success probability that is not too small compared to the success probability of the prover with fixed $x_1, \ldots, x_{i-1}$ *before appending x_i to the tree*, and then recursively call the tree-finder algorithm. More precisely, if $\varepsilon_i(x_1, \ldots, x_{i-1})$ denotes the success probability of the prover with the first $i-1$ challenges fixed to $x_1, \ldots, x_{i-1}$ if $i > 1$ (set $\varepsilon_1 = \varepsilon$), their idea is to generate a challenge x_i that is distinct from its siblings and estimate whether $\varepsilon_{i+1}(x_1, \ldots, x_i) \geq \varepsilon_i(x_1, \ldots, x_{i-1})/\alpha$ for some constant $\alpha > 1$ (to conduct their analysis, α must actually be greater than $(1-5/P)^{-2}$). To perform this estimation for a generated x_i, the tree-finder algorithm runs the protocol with the first i challenges fixed to $x_1, \ldots, x_i$ a certain number $T_i = O\left(n^{\log m + \log \alpha} T_{\mathsf{Prove}^*}/\varepsilon\right)$ of times and counts the number of successful executions. Challenge x_i is then appended to the tree only if this number is higher than a certain threshold. If the threshold is not reached, the algorithm generates a fresh challenge x_i.

Using probabilistic methods, they attempt to show that the probability that with this number T_i of protocol runs, for a certain challenge x_i, (1) the probability that $\varepsilon_{i+1} \geq \varepsilon_i/\alpha$ and that the threshold is reached is high, and (2) the probability that $\varepsilon_{i+1} < \varepsilon_i/\alpha$ although the threshold is reached is low. The second part lead to a proof that the probability that no node in the final tree is labeled with a challenge x_i such that $\varepsilon_{i+1} < \varepsilon_i/\alpha$ is high.

However, an issue in the proof of the first step is in the use of heavy-row arguments: the authors claimed that the probability that x_i is such that $\varepsilon_{i+1} \geq \varepsilon_i/\alpha$ is at least $1 - 1/\alpha$, which is not the guaranteed by heavy-row arguments. Instead, heavy-row arguments guarantee that the probability that x_i is such that $\varepsilon_{i+1} \geq \varepsilon_i/\alpha$ *conditioned on the event in which x_i leads to a success* is at least $1 - 1/\alpha$. Without the conditioning, the best known bound is only $\varepsilon_i(1 - 1/\alpha)$.

We thus propose to modify their tree-finder algorithm as follows: the algorithm does not generate a challenge x_i and start counting, but instead generates x_i and runs the protocol until the end. If the execution is not successful, then the algorithm generates a fresh challenge x_i. If the execution is successful, then the algorithm proceeds with the counting. This way, when the algorithm starts counting, x_i is already known to lead to a success and their running-time analysis applies. The expected running time to generate a value x_i that leads to a success is $T_{\mathsf{Prove}^*}/\varepsilon_i$.

Now recall that a challenge x_i is only appended to the tree if it subsequently leads to a high number of successes. The analysis of del Pino, Seiler and Lyubashevsky then continued by showing that, in expectation, a constant of x_i must be drawn before appending one to the tree, assuming that only x_i such that $\varepsilon_{i+1} \geq \varepsilon_i/\alpha$ are chosen. Therefore, the expected running time until our algorithm appends a challenge x_i to the tree is of order $O((1/\varepsilon_i + T_i)T_{\mathsf{Prove}^*})$ in expectation; and since $\varepsilon_i \geq \varepsilon/\alpha^{i-1}$, $1/\varepsilon_i = o(T_i)$, and $O((1/\varepsilon_i + T_i)T_{\mathsf{Prove}^*}) = O(T_i)$. Their analysis of the expected running time of the whole tree-finder algorithm in case only challenges such that $\varepsilon_{i+1} \geq \varepsilon_i/\alpha$ are chosen then shows that the expected time of our algorithm is of order $O\left(n^{\log m + \log \alpha} \log(n+1) T_{\mathsf{Prove}^*}/\varepsilon\right)$; and our algorithm certainly returns a challenge-tree with transcripts that are all accepting, not just with probability 1/4 as theirs.

5 Succinct Arguments for Multi-Integer Commitments

This section gives several succinct protocols related to multi-integer commitments. The first protocol allows to succinctly argue knowledge of an opening to a multi-integer commitment, and is based on the same halve-then-recurse techniques as in Section 4. Then comes a protocol that allows to aggregate arguments of knowledge of openings to several commitments (and the techniques used in it can be applied to additively-homomorphic commitment scheme in public-order groups). With the same aggregation techniques, we then show how to obtain short parameters for the commitment scheme in Section 3.2 and the inner-product argument in Section 4. Finally, we show how to succinctly argue knowledge of the same opening to several commitments in different bases.

5.1 Succinct Arguments of Openings

It is worth noting that the halving techniques used in protocol Π' can also be used to argue knowledge of openings to the multi-integer commitments presented in Section 3.2. To argue knowledge of integers $a_1, \ldots, a_n, r$ (with $n \geq 2$) of absolute value less than 2^ℓ and such that $C^2 = (\mathbf{g}^{\mathbf{a}} h^r)^4$ for positive integers n and ℓ, group elements $h, g_1, \ldots, g_n, C \in \mathbb{G}$ and a proof π_{crs} that $\mathbf{g} \in \sqrt{\langle h^2 \rangle}^n$, the prover starts by verifying π_{crs} and aborts if it is invalid. Next, she computes $U \leftarrow {\mathbf{g}_1}^{\mathbf{a}_2} h^{s_U}$ and $V \leftarrow {\mathbf{g}_2}^{\mathbf{a}_1} h^{s_V}$ for s_U and s_V uniformly random over the same range as in protocol Π', and sends them to the verifier. The verifier chooses $x \leftarrow_\$ [\![0; P-1]\!]$, sends it to the prover, and this latter computes $\mathbf{a}' \leftarrow \mathbf{a}_1 + x\mathbf{a}_2$. They then recurse (without verifying π_{crs}) with $\lceil n/2 \rceil$ as new vector-length, $\mathbf{g}_1^x \circ \mathbf{g}_2$ and h as new bases, $U^{x^2} C^x V$ as commitment, and the prover uses $\mathbf{a}'$ and $s_v + rx + s_U x^2$ as new witness. When the vector length is 1, they run the protocol in Section 3.2.

As in the inner-product protocol, the bit communication complexity of the prover is of the order of $O\left(\ell + \log(n) b_{\mathbb{G}}\right)$ bits.

This should compared with the complexity of the straightforward protocol which consists in adapting the protocol in Section 3.2 with vector of length 1 to the case of vectors of length n. The bit complexity of this latter protocol is of order $O\left(b_{\mathbb{G}} + n\left(\ell + \lambda + \log P\right)\right)$, and $O\left(n\left(\ell + \lambda + \log P\right)\right)$ with the Fiat–Shamir heuristic.

5.2 Aggregating Arguments of Openings to Integer Commitments

This section shows how to aggregate several arguments of knowledge of openings to integer commitments, i.e., given m commitments $C_1, \ldots, C_m$, how to argue *at once* knowledge of integer vectors $\mathbf{a}_j \in \mathbb{Z}^n$ and integers r_j such that $C_j^2 = (\mathbf{g}^{\mathbf{a}_j} h^{r_j})^4$ for all $j \in [\![m]\!]$. These techniques can also be applied to Pedersen's commitments in public-order groups, in which case the size of the proof is constant in m (the only effect of m is on the extraction probability).

More formally, the protocol is for the relation

$$\Big\{(\mathbf{C} \in \mathbb{G}^m, \ell \in \mathbb{N}^*; \mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}^n, \mathbf{r} \in \mathbb{Z}^m) : \forall j \in [\![m]\!]\ C_j^2 = (\mathbf{g}^{\mathbf{a}_j} h^{r_j})^4 \\ \wedge \left\|\begin{bmatrix}\mathbf{a}_j & r_j\end{bmatrix}\right\|_\infty < 2^\ell\Big\},$$

given parameters $(\mathbb{G}, P, h, n, m)$ and $(\mathbf{g}, \pi_{crs}) \in \mathbb{G}^n \times \{0,1\}^*$ as CRS. The protocol satisfies culpable extractability w.r.t. the language define similarly but without the bounds on the witness.

The construction is generic in the sense that it builds upon any protocol for a single commitment. Of course, aggregating arguments is only interesting if it leads to savings in terms of communication size and computational costs compared to m parallel executions of the protocol for a single inner product. Applied to the protocol in Section 4.1, the number of group elements sent by the prover in the aggregated argument is $2\lceil\log(n)\rceil + 1$ instead of $m(2\lceil\log n\rceil + 1)$ group elements for m parallel executions of the protocol, but the last two integers in the aggregated argument are mP times larger. Moreover, the verification of the aggregated argument requires a single multi-exponentiation instead of m, but with exponents that are mP times larger than for m separate multi-exponentiations.

Protocol. The main building block is a protocol Π for the relation in Section 5.2. At the beginning of the protocol, the verifier chooses integers $\xi_1, \ldots, \xi_m \leftarrow_\$ [\![0; P-1]\!]$ and sends them to the prover. The prover and the verifier then compute $C_1^{\xi_1} \cdots C_m^{\xi_m}$. With $(\mathbb{G}, P, h, n)$ as parameters, and $\mathbf{g}$ and the proof that $\mathbf{g} \in \sqrt{\langle h^2\rangle}^n$ as CRS, the parties run Π on the input of $C_1^{\xi_1} \cdots C_m^{\xi_m}$ and $\ell + \lfloor\log(mP)\rfloor + 1$ as maximum length (if ℓ is the maximum bit length of the integers in the witness); and the prover uses $\xi_1\mathbf{a}_1 + \cdots + \xi_m\mathbf{a}_m \in \mathbb{Z}^n$ and $\xi_m r_1 + \cdots + \xi_m r_m \in \mathbb{Z}$ as witness.

The underlying idea is simple: if the prover indeed knows openings to all commitments, then this latter should be able to open random linear combination of $C_1, \ldots, C_m$, and this is enough to convince the verifier as the prover can guess the combination $\xi_1, \ldots, \xi_m$ in advance with only negligible probability.

Alternatively, the verifier could send a single integer $\xi \leftarrow_\$ [\![0; P-1]\!]$, which would define a vector $\Xi := \begin{bmatrix}1 & \xi & \cdots & \xi^{m-1}\end{bmatrix}$. Although this would reduce the communication from the verifier to the prover, the integers in the witness would be P^m times larger instead of mP times. In case optimizing the communication size from the prover is more important (e.g., for non-interactive arguments with the Fiat–Shamir heuristic, although the computational cost of the prover increases), the first variant is preferable. It should be noted that in public-order groups, one should rather favor the second variant since the integers can always be reduced modulo the group order.

Completeness and Security. The protocol is complete by construction. Moreover, if Π is $(T, T_{\mathsf{Sim}}, \varepsilon)$-honest-verifier zero-knowledge, then so is the protocol.

As for extractability, note that for any matrix $\Xi \in \mathbb{Z}^{m\times m}$, denoting by $\mathbf{e}_j \in \mathbb{Z}^m$ the canonical row vector with 1 at position j and 0 elsewhere for $j \in [\![m]\!]$, and by $\operatorname{adj}(\Xi)$ the adjugate matrix of Ξ, the vector $\mathbf{x}_j := \mathbf{e}_j \operatorname{adj}(\Xi)$ satisfies $\mathbf{x}_j\Xi = \mathbf{e}_j \operatorname{adj}(\Xi)\Xi = \det(\Xi)\mathbf{e}_j$ since $\operatorname{adj}(\Xi)\Xi = \det(\Xi)\mathbf{I}_m$. Assuming that for all $i \in [\![m]\!]$, the equality $\left(C_1^{\xi_{i,1}} \cdots C_m^{\xi_{i,m}}\right)^2 = (\mathbf{g}^{\mathbf{a}_i} h^{r_i})^4$ holds for some integer vectors $\mathbf{a}_i \in \mathbb{Z}^m$ and integers r_i, it follows that $C_j^{2\det\Xi} = \left(\mathbf{g}^{\sum_{i=1}^m \mathbf{x}_{j,i}\mathbf{a}_i} h^{\sum_{i=1}^m \mathbf{x}_{j,i} r_i}\right)^4$ for all $j \in [\![m]\!]$.

Consider then an extractor which first generates $\Xi_1 \leftarrow_\$ [\![0; P-1]\!]^m$. If $\Xi_1 = 0_{\mathbb{Z}^m}$, then it generates a new vector Ξ_1 and otherwise runs the protocol with Prove^* with Ξ_1 as first message from the verifier. If the protocol execution fails, then the extractor rewinds the Prove^* to the beginning of the protocol and generates a fresh vector Ξ_1.

Note that conditioned on the event in which $\Xi_1 \neq 0_{\mathbb{Z}^m}$, Prove^* convinces the verifier with probability at least $\varepsilon - P^{-m}$. Moreover, conditioned on the first message Ξ_1, a heavy-row argument implies that with probability at least 1/2, the row vector Ξ_1 is such that the verifier of the sub-protocol Π is convinced with probability at least $(\varepsilon - P^{-m})/2$ on input $C_1^{\xi_{1,1}} \cdots C_m^{\xi_{1,m}}$.

The extractor then runs $\Pi.\mathcal{E}$ on Prove^* from the computation step after it has received Ξ_1. If $\Pi.\mathcal{E}$ does not return a value in at most twice the expected value of its running time with a prover that succeeds with probability at least $(\varepsilon - P^{-m})/2$, the extractor generates a new vector Ξ_1 and proceeds as before. Denote by H (as in "heavy") the event in which Ξ_1 is such that the verifier of Π is convinced with probability at least $(\varepsilon - P^{-m})/2$, and by T the event in which $\Pi.\mathcal{E}$ returns a value in at most twice the expected value of its running time with a prover that succeeds with probability at least $(\varepsilon - P^{-m})/2$ for a given $C_1^{\xi_{1,1}} \cdots C_m^{\xi_{1,m}}$ (Markov's inequality implies that this event occurs with probability at most 1/2). The extractor returns a value in the event $H \cap T$ and $\Pr[T \cap H] = \Pr[T|H]\Pr[H] \geq 1/4$. Therefore, the generator restarts from the generation of Ξ_1 an expected number of times at most 4 and at each repetition, its running time is at most $\tilde{\varepsilon}^{-1}(1-P^{-m})^{-1}T_{\mathsf{Prove}^*} + 2T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)$, with $\tilde{\varepsilon} := \varepsilon - P^{-m}$ and $T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)$ the expected running time of $\Pi.\mathcal{E}$ given a prover that succeeds with probability at least $\tilde{\varepsilon}/2$. The term T_{Prove^*} comes from the fact that the generator must determine whether Ξ_1 leads to a success. The expected running time is at most $4\left(\tilde{\varepsilon}^{-1}(1-P^{-m})^{-1}T_{\mathsf{Prove}^*} + 2T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)\right)$.

Now, for $i \in \{2, \ldots, m\}$, the extractor generates $\Xi_i \leftarrow_\$ [\![0; P-1]\!]^m$. If $\Xi_1, \ldots, \Xi_i$ are linearly dependent over $\mathbb{Q}$, then the extractor generates a new vector $\Xi_i \leftarrow_\$ [\![0; P-1]\!]^m$. Slinko [33, Corollary 2] proved that this event occurs with probability at most P^{-m+i-1}. It follows that for fixed $\Xi_1, \ldots, \Xi_{i-1}$, conditioned on the event in which $\Xi_1, \ldots, \Xi_i$ are linearly independent, Prove^* convinces the verifier with probability at least $\varepsilon - P^{-m+i-1}$. The extractor proceeds as in the case $i = 1$ but with $P^{-m+i-1} \leq P^{-1}$ instead of P^{-m}. Note, however, that the extractor must determine whether $\Xi_1, \ldots, \Xi_i$ are linearly independent, and it can do so by computing its rank, i.e., the rank of a matrix with coefficient in $[\![0; P-1]\!]$. Using Bareiss algorithm [3] which requires $O(m^3)$ arithmetic operations on integers less than $O\left(m^{m/2}P^m\right)$, i.e., it can be done in time $O\left(m^5(\log m + \log P)^2\right)$. Besides, the extractor must determine whether Ξ_i leads to a success, which takes time at most T_{Prove^*}. At any step $i \in [\![m]\!]$, the expected running time is thus at most $4\left(\tilde{\varepsilon}^{-1}O\left(\left(1-P^{-1}\right)^{-1}\left(m^5(\log m + \log P)^2 + T_{\mathsf{Prove}^*}\right)\right) + 2T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)\right)$ for $\tilde{\varepsilon} := \varepsilon - P^{-1}$.

Consequently, the extractor can obtain vectors $\mathbf{a}_i \in \mathbb{Z}^n$ and integers r_i such that $\left(C_1^{\xi_{i,1}} \cdots C_m^{\xi_{i,m}}\right)^2 = (\mathbf{g}^{\mathbf{a}_i} h^{r_i})^4$ for all $i \in [\![m]\!]$ and for some $\Xi \in \mathbb{Z}^{m \times m}$ such that $\det \Xi \neq 0$ in expected time at most

$$4m\left(O\left(\tilde{\varepsilon}^{-1}\left(1-P^{-1}\right)^{-1}\left(m^5(\log m + \log P)^2 + T_{\mathsf{Prove}^*}\right)\right) + 2T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)\right).$$

Lemma 4.4 then shows that with probability at least

$$1 - \left(1/2 - 2^{-\lambda} - (1-\mu)\right)^{-1}\left(\varepsilon^{\mathrm{ord}} + \varepsilon^{\mathrm{strg}} + 1 - \mu\right),$$

$\det \Xi$ divides $2\sum_{i=1}^m \mathrm{x}_{i,j}\mathbf{a}_i$ and $2\sum_{i=1}^m \mathrm{x}_{i,j}r_i$ for all $j \in [\![m]\!]$ under the $(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}})$-strong-root assumption, the $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$-small-order assumption, the low-dyadic-valuation assumption and the μ-assumption over G, if $T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)$ plus the time to compute $2\sum_{i=1}^m \mathrm{x}_{i,j}\mathbf{a}_i$ and $2\sum_{i=1}^m \mathrm{x}_{i,j}r_i$, which is denoted T, is such that $n(b_{\mathbb{G}} + \log n)TT_{\mathbb{G}} + (T + b_{\mathbb{G}} + \log n)\log(P)T_{\mathbb{G}} \leq \Omega\left(\min\left(T^{\mathrm{strg}}, T^{\mathrm{ord}}\right)\right)$. The coefficient of $\mathrm{adj}(\Xi)$ are of order $O(mP^m)$ in absolute value, and the components of $\mathbf{a}_i$ and r_i are at most $2^{O(T_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2))}$ in absolute value. Therefore, $2\sum_{i=1}^m \mathrm{x}_{i,j}\mathbf{a}_i$ and $2\sum_{i=1}^m \mathrm{x}_{i,j}r_i$ can be computed in time $O\left(nmT_{\Pi.\mathcal{E}}(\tilde{\varepsilon}/2)(\log m + m\log P)\right)$.

5.3 Shorter Parameters for Integer Commitments

The keys of the multi-integer commitment scheme in Section 3.2 include a proof that that they were well-formed, i.e., that if the scheme allows to commit to m integers, then the group elements $g_1, \ldots, g_m$ in the key are all in $\sqrt{\langle h^2 \rangle}$ for $h \leftarrow_{\$} \mathbb{G}$. One can of course run m times in parallel the protocol in Section 3.2, which would result in arguments of $O(mb_{\mathbb{G}})$ bits. Alternatively, one could use the same techniques as in Section 5.2 to aggregate these arguments and obtain a single argument of $O(b_{\mathbb{G}} + \log m)$ bits.

In more detail, the protocol is a proof system for the language

$$\left\{g_1, \ldots, g_m \in \mathbb{G}, \ell \in \mathbb{N}^* : \exists \alpha_1, \ldots, \alpha_m \in \left[\!\left[0; 2^{\ell}\right]\!\right], \forall i \in [\![m]\!]\; g_i = h^{\alpha_i}\right\},$$

given parameters $(\mathbb{G}, P, h, m)$ and the empty string as CRS. It guarantees that $g_i \in \sqrt{\langle h^2 \rangle}$ for all $i \in [\![m]\!]$. At the beginning of the protocol, the verifier chooses integers $\xi_1, \ldots, \xi_m \leftarrow_{\$} [\![0; P-1]\!]$ and sends them to the prover. The prover and the verifier compute $g_1^{\xi_1} \cdots g_m^{\xi_m}$, and run the protocol of Section 3.2 (i.e., for the case $m = 1$) on the input of $g_1^{\xi_1} \cdots g_m^{\xi_m}$ and $\ell + \lfloor \log(mP) \rfloor + 1$ as maximum bit length; and the prover uses $\xi_1 x_1 + \cdots + \xi_m x_m \in \mathbb{Z}$ as witness. With the Fiat–Shamir heuristic, the proof then consists of $2\lfloor \log P \rfloor + \ell + \lfloor \log(mP) \rfloor + \lambda + 4$ bits. For $\ell = b_{\mathbb{G}} + \lambda$, that is $O(b_{\mathbb{G}} + \log m)$ bits (recall that $P \leq 2^{b_{\mathbb{G}}}$ and $b_{\mathbb{G}} = \Omega(\lambda)$).

The same arguments as in Section 5.2 imply that this protocol is complete, statistically honest-verifier zero-knowledge and extractable.

Similarly, the CRS of the inner-product protocol in Section 4.1 includes an argument that the bases $\mathbf{g}$ and $\mathbf{h} \in \mathbb{G}^n$ (for some $n \in \mathbb{N}^*$) are in $\sqrt{\langle f^2 \rangle}^n$ for $f \leftarrow_\$ \mathbb{G}$. The same technique can then be used to obtain an argument of $O(b_\mathbb{G} + \log n)$ bits instead of $O(nb_\mathbb{G})$ bits.

5.4 Succinct Base-Switching Arguments

This section shows how to succinctly argue knowledge of an integer vector $\mathbf{a} \in \mathbb{Z}^n$ and of integers $r_1, \ldots, r_m$ such that $(\mathbf{a}, r_1), \ldots, (\mathbf{a}, r_m)$ respectively open to commitments $C_1, \ldots, C_m$ w.r.t. bases $(\mathbf{g}_1, h), \ldots, (\mathbf{g}_m, h) \in \mathbb{G}^{n+1}$. Said otherwise, the same vector $\mathbf{a}$ is committed in m different bases. Formally, the protocol is for the relation

$$\left\{(\mathbf{C} \in \mathbb{G}^m, \ell \in \mathbb{N}^*; \mathbf{a} \in \mathbb{Z}^n, \mathbf{r} \in \mathbb{Z}^m) : \forall j \in [\![m]\!]\ C_j^2 = \left(\mathbf{g}_j{}^{\mathbf{a}} h^{r_j}\right)^4 \wedge \left\|\left[\mathbf{a}\ \mathbf{r}\right]\right\| < 2^\ell\right\},$$

given parameters $(\mathbb{G}, P, h, n, m)$ and $(\mathbf{g}_1, \ldots, \mathbf{g}_m, \pi_{crs}) \in \mathbb{G}^{m \times n} \times \{0,1\}^*$ as CRS, and π_{crs} is an argument that each of the components of all the $\mathbf{g}_j$ vectors are in $\sqrt{\langle h^2 \rangle}$.

The idea underlying the protocol is again to use linear combinations to reduce $C_1, \ldots, C_m$ to a single group element, and run the succinct argument for multi-integer openings on the linear combination of the basis. More precisely, the verifier generates $\xi \leftarrow_\$ [\![0; P-1]\!]^m$ and sends it to the prover. Both parties compute $C \leftarrow \mathbf{C}^\xi = C_1^{\xi_1} \cdots C_m^{\xi_m}$ and $\mathbf{g} \leftarrow \mathbf{g}_1^{\xi_1} \circ \cdots \circ \mathbf{g}_m^{\xi_m}$. They then run the protocol for multi-integer openings on the input of C and $\mathbf{g}$, and the prover uses $\mathbf{a}$ and $\xi_1 r_1 + \cdots + \xi_m r_m$ as witness. The new witness is then at most mP times larger than the original one.

The completeness and the zero-knowledge property of the scheme are immediate. Concerning its extractability, notice that if for all $i \in [\![m]\!]$ one can obtain vectors $\mathbf{a}_i \in \mathbb{Z}^m$ and integers r_j such that $\mathbf{C}^{2\xi_i} = \left(g_1^{\xi_{i,1}} \circ \cdots \circ \mathbf{g}_m^{\xi_{i,m}}\right)^{4\mathbf{a}_i} h^{4r_i}$, with $\xi_1, \ldots, \xi_m$ vectors in $[\![0; P-1]\!]^m$ that are linearly independent over $\mathbb{Q}$, then one can first compute $\mathbf{x}_j \coloneqq \mathbf{e}_j \operatorname{adj}(\Xi)$, where $\mathbf{e}_j \in \mathbb{Z}^m$ denotes the canonical row vector with 1 at position j and 0 elsewhere for all $j \in [\![m]\!]$, Ξ denotes the matrix with rows $\xi_1, \ldots, \xi_m$ and $\operatorname{adj}(\Xi)$ its adjugate. Then, since $\mathbf{x}_j \Xi = \det(\Xi)\mathbf{e}_j$, it follows that for all $j \in [\![m]\!]$,

$$\begin{aligned}
C_j^{2\det\Xi} &= \prod_{i=1}^m \left(\mathbf{g}_1{}^{\xi_{i,1}} \circ \cdots \circ \mathbf{g}_m{}^{\xi_{i,m}}\right)^{4\mathbf{x}_{j,i}\mathbf{a}_j} h^{\sum_i 4\mathbf{x}_{j,i} r_j} \\
&= \prod_{i=1}^m \left(\mathbf{g}_1{}^{\mathbf{x}_{j,i}\xi_{i,1}} \circ \cdots \circ \mathbf{g}_m{}^{\mathbf{x}_{j,i}\xi_{i,m}}\right)^{4\mathbf{a}_j} h^{\sum_i 4\mathbf{x}_{j,i} r_j} \\
&= \left(\mathbf{g}_1{}^{\sum_i \mathbf{x}_{j,i}\xi_{i,1}} \circ \cdots \circ \mathbf{g}_m{}^{\sum_i \mathbf{x}_{j,i}\xi_{i,m}}\right)^{4\mathbf{a}_j} h^{\sum_i 4\mathbf{x}_{j,i} r_j} \\
&= \mathbf{g}_j{}^{4\det(\Xi)\mathbf{a}_j} h^{\sum_i 4\mathbf{x}_{j,i} r_j}.
\end{aligned}$$

The second and third equalities respectively rely on the fact that for any two vectors $\mathbf{e}, \mathbf{f} \in \mathbb{G}^n$ and any vector $\mathbf{c} \in \mathbb{Z}^n$, $(\mathbf{e} \circ \mathbf{f})^{\mathbf{c}} = \mathbf{e}^{\mathbf{c}}\mathbf{f}^{\mathbf{c}}$, and for any two integers c, d, $\mathbf{e}^c \circ \mathbf{e}^d = \mathbf{e}^{cd}$. Using rewinding techniques similar to those in the extractability proof the aggregated arguments then shows that the protocol is extractable.

6 Succinct Argument for Diophantine Equations

This sections gives a succinct argument to argue satisfiability of Diophantine equations. Although Davis, Putnam, Robinson and Matiyasevich [29] showed that there does not exist an algorithm that can decide whether any Diophantine equation has a solution, thereby giving a negative answer to Hilbert's tenth problem; one can argue in zero-knowledge knowledge of a solution, if a solution is known to the prover, which convinces the verifier that the equation is satisfiable.

Damgård and Fujisaki gave [15, Section 4.2] a protocol to argue, given three commitments C_1, C_2, C_3 computed with their scheme, knowledge of openings x_1, x_2, x_3 such that $x_3 = x_1 x_2$. Therefore, to show the satisfiability of an ν-variate polynomial $\sum_{\mathbf{i} \in \mathbb{N}^\nu} a_{\mathbf{i}} x_1^{i_1} \cdots x_\nu^{i_\nu}$ of total degree δ using their scheme, if the polynomial can be computed in $M(\nu, \delta)$ multiplications, then one would have to compute $2M(\nu, \delta) + 1$ integer commitments and compute $M(\nu, \delta)$ multiplication-consistency arguments. As Damgård and Fujisaki's scheme is additively homomorphic, the verifier can verify addition itself.

Computing a monomial $x_1^{i_1} \cdots x_\nu^{i_\nu}$ can be done in at most $\delta - 1$ multiplications since the polynomial is of total degree δ. Without any further restriction on the polynomial than its number of variables ν and its total degree δ, the best bound on the number of multiplications (between variables) one can give is $\delta - 1$ as δ could be less than ν, and all i_k at most 1. Evaluating an ν-variate polynomial of total degree δ thus *a priori* requires $(\delta - 1)\binom{\nu+\delta}{\delta}$ multiplications as such a polynomial has at most $\binom{\nu+\delta}{\delta}$ monomials. This can be improved to $\binom{\nu+\delta}{\delta} - \nu - 1 \leq \binom{\nu+\delta}{\delta}$ multiplications by evaluating all possible monomials (even those which may have coefficient 0) recursively by increasing degree and storing the previous evaluations. There exist more efficient methods for specific polynomials (e.g., recursive Horner's method or summation a few multi-exponentiations for polynomials with a small numbers of monomials of large degree) but no better upper-bound on the number of multiplications is known for a generic polynomial.

Consider a prover that wants to argue the satisfiability of a (generic) ν-variate polynomial of total degree δ with integer coefficients whose absolute value is upper-bounded by 2^H for some integer H. The communication complexity of the arguments of the first multiplication gates are order $\Omega(\log P + \ell + b_{\mathbb{G}})$ if ℓ denotes the maximum bit length of any coordinate in the solution. Since the total degree of the polynomial is δ, the bit length of the witness at the maximum-depth multiplication gates can be as large as $\delta\ell + \log\left(\binom{\nu+\delta}{\delta}\right) H$ and the communication complexity of the argument of the satisfiability of the Diophantine equation (i.e., the proof that the polynomial actually evaluates to 0) is $\Omega\left(\delta\ell + \log\left(\binom{\nu+\delta}{\delta}\right) H + b_{\mathbb{G}}\right)$. The overall communication complexity with Damgård and Fujisaki's scheme is therefore upper-bounded by $O\left(\binom{\nu+\delta}{\delta}\left(\delta\ell+\log\left(\binom{\nu+\delta}{\delta}\right)H+b_{\mathbb{G}}\right)\right)$ and lower-bounded by $\Omega\left(\binom{\nu+\delta}{\delta}(\ell+b_{\mathbb{G}})\right)$ for generic polynomials.

This section shows how to argue the satisfiability of Diophantine equations with a communication complexity of order $O\left(\delta\ell+\min(\nu,\delta)\log\left(\nu+\delta\right)b_{\mathbb{G}}+H\right).$

6.1 Arguments via Polynomial-Degree Reductions

Our approach to argue for Diophantine satisfiability is different and is inspired by Skolem's method [32]. The idea is to give a systematic method to turn any polynomial equation to another of degree at most 4 by increasing the number of variables so that the satisfiability of one polynomial implies that of the other. The resulting polynomial is such that its satisfiability is equivalent to the satisfiability (over the integers) of a Hadamard product of the form $\mathbf{a}_L\circ\mathbf{a}_R=\mathbf{a}_O$ and of linear equations with the entries of $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O$ as indeterminate. The length of these latter vectors is the number of variables in the resulting polynomial, and if the original polynomial is ν-variate and of total degree at most δ, then the new polynomial has at most $\nu\lfloor\log\delta\rfloor+(\delta-1)\mu$ variables, where $\mu\leq\binom{\nu+\delta}{\delta}$ is the number of monomials in the original polynomial.

On this account, if one can argue for the satisfiability of such Hadamard products and linear constraints, then one can argue for the satisfiability of the original polynomial. In the protocol given in Section 6.2, the prover only sends logarithmically many group elements in the length of the vectors in the Hadamard product, and a constant number of integers. The bit length of those integers is upper-bounded by $O\left(\delta\ell+b_{\mathbb{G}}+\min(\nu,\delta)\log\left(\nu+\delta\right)\log P+H\right)$ if the bit length of the witness is upper-bounded by ℓ and the bit length of each coefficient of the polynomial is at most H.

Reducing Arbitrary Polynomials to Polynomials of Degree at most 4. We now give a systematic procedure to reduce any Diophantine equation into an equation of degree at most 4 of which the satisfiability can be reduced to the satisfiability of a Hadamard product and linear constraints; and the Hadamard product and the constraints can be read immediately from the resulting polynomial. The presentation is gradual as it starts with ν-variate affine equations, proceeds with ν-variate Diophantine equations in which the degree in each variable is at most 1, further tackles univariate polynomials of arbitrary degree and then considers arbitrary Diophantine equations. The method applies to every multivariate integer polynomial, but for specific polynomials, more astute techniques could lead to a smaller number of new variables and/or constraints.

Step 1–Affine Equations. Given an integer polynomial $a_1x_1+\cdots+a_\nu x_\nu+b\in\mathbb{Z}[x_1,\ldots,x_\nu]$, set $\mathbf{a}_O\leftarrow\begin{bmatrix}x_1\cdots x_\nu\end{bmatrix}$ and for all $i\in[\![\nu]\!]$, set $\mathbf{a}_{L,i}=1$ and $\mathbf{a}_{R,i}=x_i$. The equation $a_1x_1+\cdots+a_\nu x_\nu+b=0$ is satisfied if and only if $\left\langle\begin{bmatrix}a_1\cdots a_\nu\end{bmatrix},\mathbf{a}_O\right\rangle=-b$ and $\mathbf{a}_L\circ\mathbf{a}_R=\mathbf{a}_O$. Note that no variable or linear constraint was added to the system of equations.

Step 2–Restricted Diophantine Equations. Consider an integer polynomial $\sum_{\mathbf{i}\in\mathbb{N}^\nu} a_{\mathbf{i}} x_1^{i_1}\cdots x_\nu^{i_\nu} \in \mathbb{Z}[x_1,\ldots,x_\nu]$ of total degree δ such that $a_{\mathbf{i}} \neq 0_{\mathbb{Z}} \implies \mathbf{i} \in \{0,1\}^\nu$, i.e., the polynomial is of degree at most 1 in each variable. For all $\mathbf{i} \in \mathbb{N}^\nu \setminus \{0_{\mathbb{N}^\nu}\}$ such that $a_{\mathbf{i}} \neq 0_{\mathbb{Z}}$, let $\{j_1,\ldots,j_{w(\mathbf{i})}\}$ be the subset of $[\![\nu]\!]$ such that $j_1 < \cdots < j_{w(\mathbf{i})}$ and $i_{j_1} = \cdots = i_{w(\mathbf{i})} = 1$, with $w(\mathbf{i})$ denoting the Hamming weight of $\mathbf{i}$ (which is necessarily less than δ). If $w(\mathbf{i}) > 1$, introduce new variables

$$u_{\mathbf{i},1} \leftarrow x_{j_1}x_{j_2}, \quad u_{\mathbf{i},2} \leftarrow u_{\mathbf{i},1}x_{j_3}, \quad \ldots, \quad u_{\mathbf{i},w(\mathbf{i})-1} \leftarrow u_{\mathbf{i},w(\mathbf{i})-2}x_{j_{w(\mathbf{i})}},$$

with the convention that $u_{\mathbf{i},0} \coloneqq x_{j_1}$. Note that $\sum_{\mathbf{i}\in\mathbb{N}^\nu} a_{\mathbf{i}} x_1^{i_1}\cdots x_\nu^{i_\nu} = 0$ if and only if

$$\sum_{\substack{\mathbf{i}\in\mathbb{N}^\nu \,:\, a_{\mathbf{i}}\neq 0_{\mathbb{Z}}\\ w(\mathbf{i})>1}} \sum_{k=1}^{w(\mathbf{i})-1} \left(u_{\mathbf{i},k} - u_{\mathbf{i},k-1}x_{j_{k+1}}\right)^2 + \left(\sum_{\mathbf{i}\in\mathbb{N}^\nu} \mathbf{a}_{\mathbf{i}} u_{\mathbf{i},w(\mathbf{i})-1}\right)^2 = 0,$$

with the convention that $u_{\mathbf{0}_{\mathbb{N}^\nu},-1} = 1$. This latter polynomial is of degree 4, and the equation is satisfied if and only if the linear equation $\sum_{\mathbf{i}\in\mathbb{N}^\nu} \mathbf{a}_{\mathbf{i}} u_{\mathbf{i},w(\mathbf{i})-1} = 0$ is as well as the constraints $u_{\mathbf{i},k} - u_{\mathbf{i},k-1}x_{j_{k+1}} = 0$. Set then

$$\begin{aligned}
\mathbf{a}_L &\leftarrow \left[x_{j_1}\ u_{\mathbf{i},1}\ \cdots\ u_{\mathbf{i},w(\mathbf{i})-2}\right]\\
\mathbf{a}_R &\leftarrow \left[x_{j_2}\ x_{j_3}\ \cdots\ x_{j_{w(\mathbf{i})}}\right]\\
\mathbf{a}_O &\leftarrow \left[u_{\mathbf{i},1}\ u_{\mathbf{i},2}\ \cdots\ u_{\mathbf{i},w(\mathbf{i})-1}\right],
\end{aligned}$$

and introduce the linear constraints $\mathbf{a}_{L,i+1} - \mathbf{a}_{O,i} = 0$ for $i \in \{1,\ldots,w(\mathbf{i})-2\}$. The procedure introduces at most $\delta - 1$ new variables and $\delta - 2$ new linear constraints per monomial, and since there are at most $\binom{\nu+\delta}{\delta}$ monomials in an ν-variate polynomial of total degree δ, that is at most $(\delta-1)\binom{\nu+\delta}{\delta}$ variables and $(\delta-2)\binom{\nu+\delta}{\delta}$ constraints.

Step 3–Univariate Polynomials. Given a polynomial $Z = a_0 + a_1x + \cdots + a_\delta x^\delta \in \mathbb{Z}[x]$ of degree $\delta \geq 2$, introduce variables

$$u_1 \leftarrow x^2, \quad u_2 \leftarrow u_1^2, \quad \ldots, \quad u_{\lfloor\log\delta\rfloor} \leftarrow u_{\lfloor\log\delta\rfloor-1}^2.$$

Now notice that $a_0 + a_1x + \cdots + a_\delta x^\delta = 0$ if and only if

$$\left(u_1 - x^2\right)^2 + \sum_{i=2}^{\lfloor\log\delta\rfloor} \left(u_i - u_{i-1}^2\right)^2 + \left(Z'(x,u_1,\ldots,u_{\lfloor\log\delta\rfloor})\right)^2 = 0,$$

where $Z'(x,u_1,\ldots,u_{\lfloor\log\delta\rfloor})$ is $\lfloor\log\delta\rfloor + 1$-variate integer polynomial in which the degree of each variable is at most 1, i.e., if and only if $Z'(x,u_1,\ldots,u_{\lfloor\log\delta\rfloor}) = 0$ and the constraints $u_1 - x^2 = 0$ and $u_{i+1} - u_i^2 = 0$ are satisfied.

Since

$$\sum_{i=0}^{\delta} a_ix^i = a_0 + \sum_{k=0}^{\lfloor\log\delta\rfloor} \sum_{i=2^k}^{2^{k+1}-1} a_ix^i = a_0 + \sum_{k=0}^{\lfloor\log\delta\rfloor} \sum_{i=2^k}^{2^{k+1}-1} a_ix^{i_0}u_1^{i_1}\cdots u_{k-1}^{i_{k-1}}u_k,$$

where $i_0, \dots, i_{k-1}$ is the binary decomposition of i and $a_i := 0$ for $i > \delta$, this give an explicit expression for Z'.

Set then $\mathbf{a}_L \leftarrow \mathbf{a}_R \leftarrow \left[x\ u_1 \cdots u_{\lfloor \log \delta \rfloor - 1}\right]$ and $\mathbf{a}_O \leftarrow \left[u_1\ u_2 \cdots u_{\lfloor \log \delta \rfloor}\right]$, and introduce constraints

$$\mathbf{a}_{L,i+1} - \mathbf{a}_{O,i} = \mathbf{a}_{R,i+1} - \mathbf{a}_{O,i} = 0$$

for all $i \in [\![\lfloor \log \delta \rfloor - 1]\!]$.

As the second step shows that the satisfiability of Z' can be reduced to a Hadamard product and linear constraints, the satisfiability of Z can be reduced to a Hadamard product and linear constraints. This procedure introduces $\lfloor \log \delta \rfloor$ new variables and $2\left(\lfloor \log \delta \rfloor - 1\right)$ new linear constraints. It is important for Step 4 to remark that the number of monomial of Z' is at most the same as the number of monomials in Z.

Step 4–Arbitrary Diophantine Equations. For any integer polynomial $Z = \sum_{\mathbf{i} \in \mathbb{N}^\nu} a_{\mathbf{i}} x_1^{i_1} \cdots x_\nu^{i_\nu} \in \mathbb{Z}[x_1, \dots, x_\nu]$ (for $\nu \geq 2$) of total degree δ, apply Step 3 to Z considering it as a polynomial in $\mathbb{Z}[x_2, \dots, x_\nu][x_1]$, i.e., a polynomial in x_1 with coefficients in $\mathbb{Z}[x_2, \dots, x_\nu]$. Let Z' be the resulting polynomial with coefficients in $\mathbb{Z}[x_2, \dots, x_\nu]$ and of degree at most 1 in each variable as in Step 3. Repeat Step 3 with Z' and variable x_2. After Step 3 has been repeated for each $x_1, \dots, x_\nu$, at most $\nu \lfloor \log \delta \rfloor$ new variables and $2\nu(\lfloor \log \delta \rfloor - 1)$ new linear constraints have been introduced, the resulting polynomial is of degree at most 1 in all variables and has coefficients in $\mathbb{Z}$. Concerning its total degree, note that during the process, for each monomial $x_1^{i_1} \cdots x_\nu^{i_\nu}$, the term $x_k^{i_k}$ is replaced by at most one variable if $i_k \leq 2$ and by the product of $\log i_k + 1 \leq i_k$ variables if $\text{ß}_k > 2$ for all $k \in [\![\nu]\!]$, so the total degree remains at most δ. Now apply then Step 2 to the resulting polynomial.

In summary, the procedure reduces the satisfiability of any polynomial in $\mathbb{Z}[x_1, \dots, x_\nu]$ of total degree δ with μ monomials ($\mu \leq \binom{\nu+\delta}{\delta}$ necessarily) to the satisfiability of a Hadamard product $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$, with $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O$ integer vectors of length at most $\nu \lfloor \log \delta \rfloor + (\delta - 1)\mu$, and Q linear constraints of the form

$$\langle \mathbf{w}_{L,q}, \mathbf{a}_L \rangle + \langle \mathbf{w}_{R,q}, \mathbf{a}_R \rangle + \langle \mathbf{w}_{O,q}, \mathbf{a}_O \rangle = c_q$$

for all $q \in [\![Q]\!]$ with $Q \leq 1 + 2\nu(\lfloor \log \delta \rfloor - 1) + (\delta - 2)\mu$ and with $\mathbf{w}_{L,q}, \mathbf{w}_{R,q}, \mathbf{w}_{O,q}$ integer vectors and $c_q \in \mathbb{Z}$. The coefficients of the linear constraints introduced by the procedure are in $\{-1, 0, 1\}$, except for one of which the coefficients are the coefficients of the original polynomial.

Example. As a simple illustration of the procedure, consider the polynomial $2x^3 + xy - 1$. The procedure introduces new variables $u \leftarrow x^2$, $v \leftarrow xy$ and $w \leftarrow ux$, and the equation $2x^3 + xy - 1 = 0$ is satisfiable if and only if $\left(u - x^2\right)^2 + (v - xy)^2 + (w - ux)^2 + (2w + v - 1)^2 = 0$, which allows to write a Hadamard product and linear constraints which are satisfiable if and only if this latter polynomial is.

Diophantine Equations as Circuits. It is worth noting that any polynomial in $\mathbb{Z}[x_1, \ldots, x_v]$ can naturally be viewed as an arithmetic circuit with integer inputs, and addition gates correspond to addition between two integers and similarly for multiplication gates. Nevertheless, different circuits can compute the same polynomial. For instance, the polynomial $xy + 2y = y(x + 2)$ can be computed by multiplying x and y, multiplying y by 2 and adding the result, or by adding 2 to x and multiplying the result by y. The fewer multiplication gates there are in a circuit that represents a polynomial, the smaller the communication cost of the protocol for its satisfiability will be. In any case, one can always use the circuit directly inferred by its representation as sum of monomials.

Bootle, Cerulli, Chaidos, Groth and Petit [8, Appendix A] described a procedure to turn any arithmetic circuit over $\mathbb{Z}_p$ into a circuit with only multiplication gates together with a list of linear constraints to ensure consistency between the outputs of a multiplication gate and the inputs of the gates at the next depth level of the circuit. The procedure replaces addition gates and multiplication by a constant with linear constraints and only retains multiplication gates. It ensures that the new circuit and the constraints are satisfiable if and only if the original circuit is satisfiable.

More precisely, their procedure converts any arithmetic circuit with n multiplication gates into a Hadamard product $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$, with $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O$ in $\mathbb{Z}_p^n$, and $Q \leq 2n$ linear constraints of the form

$$\langle \mathbf{w}_{L,q}, \mathbf{a}_L \rangle + \langle \mathbf{w}_{R,q}, \mathbf{a}_R \rangle + \langle \mathbf{w}_{O,q}, \mathbf{a}_O \rangle = c_q$$

for $q \in [\![Q]\!]$, with $\mathbf{w}_{L,q}, \mathbf{w}_{R,q}, \mathbf{w}_{O,q} \in \mathbb{Z}_p^n$ and $c_q \in \mathbb{Z}_p$. The vectors $\mathbf{a}_L$, $\mathbf{a}_R$ respectively denote the vectors of left and right inputs to the multiplication gates, and $\mathbf{a}_O$ the vector of outputs.

Unfortunately, their procedure cannot be directly use for circuits over the integers as it requires to re-write a linear equation $\mathbf{y} = \mathbf{A}\mathbf{x}$ (with indeterminate $\mathbf{x}$) as an equation $\mathbf{z} = \mathbf{A}'\mathbf{x}$, with $\mathbf{A}'$ in reduced row-echelon form obtained from $\mathbf{A}$ via Gaussian elimination. Computing $\mathbf{A}'$ may thus possibly require to invert entries of $\mathbf{A}$. Yet, for any matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$, there exist [1, Theorem 14.4.6] invertible integer matrices $\mathbf{Q} \in GL_m(\mathbb{Z})$ and $\mathbf{P} \in GL_n(\mathbb{Z})$ such that $\mathbf{Q}^{-1}\mathbf{A}\mathbf{P}$ is a matrix of the form $\operatorname{diag}(d_1, \ldots, d_k, 0, \ldots, 0)$, with all d_i positive integers such that $d_1 \mid d_2 \mid \cdots \mid d_k$. Given this observation, the equation $\mathbf{y} = \mathbf{A}\mathbf{x}$ can be re-written as $\mathbf{z} = \mathbf{A}'\mathbf{x}$, with $\mathbf{A}'$ of the previous form, i.e., $z_i = d_i x_i$ for $i \in [\![k]\!]$, and the circuit cannot be satisfied if $z_i \neq 0$ for any $k > i$. By introducing new variables $z_i := d_i x_i$, and increasing the number of constraints to include the constraints $z_i - d_i x_i = 0$, one could probably proceed as they did and have at most $Q \leq 3n$ constraints to satisfy.

The issue with using this procedure to argue for Diophantine satisfiability is that one cannot readily infer the constraints from the initial polynomial and one must always determine them on a case-by-case basis. Besides, if one uses the circuit directly inferred by the monomials of the polynomial without introducing new variables to decrease its degree (which would amount to modifying the circuit), computing x_1^δ for instance requires $\delta-1$ multiplications instead of $\lfloor \log \delta \rfloor$ as with our method.

6.2 Protocol

Section 6.1 shows how to reduce the satisfiability of any polynomial in $\mathbb{Z}[x_1, \ldots, x_\nu]$ of total degree δ with μ monomials ($\mu \leq \binom{\nu+\delta}{\delta}$ necessarily) to the satisfiability of a Hadamard product $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$, with $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_O$ integer vectors of length at most $\nu\lfloor \log \delta \rfloor + (\delta - 1)\mu$, and $1 + 2\nu(\lfloor \log \delta \rfloor - 1) + (\delta - 2)\mu$ linear constraints of the form

$$\langle \mathbf{w}_{L,q}, \mathbf{a}_L \rangle + \langle \mathbf{w}_{R,q}, \mathbf{a}_R \rangle + \langle \mathbf{w}_{O,q}, \mathbf{a}_O \rangle = c_q$$

for all $q \in [\![Q]\!]$, with $\mathbf{w}_{L,q}, \mathbf{w}_{R,q}, \mathbf{w}_{O,q}$ integer vectors and $c_q \in \mathbb{Z}$.

To argue for Diophantine satisfiability, it thus suffices to give a protocol protocol such relations. The following protocol is actually for more general relations in which variables of the polynomial can be committed (with the scheme in Section 3), which allows to argue on committed values while saving the cost of encoding the commitment scheme as an integer polynomial. More precisely, the protocol is for the relation

$$\Big\{\Big(\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}^{Q\times n}, \mathbf{W}_V \in \mathbb{Z}^{Q\times m}, \mathbf{V} \in \mathbb{G}^m, \mathbf{c} \in \mathbb{Z}^Q, \ell \in \mathbb{N}^*; \mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{Z}^n, \mathbf{v}, \rho \in \mathbb{Z}^m\Big) : \\ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \wedge \mathbf{W}_L \mathbf{a}_L^{\mathrm{T}} + \mathbf{W}_R \mathbf{a}_R^{\mathrm{T}} + \mathbf{W}_O \mathbf{a}_O^{\mathrm{T}} = \mathbf{W}_V \mathbf{v}^{\mathrm{T}} + \mathbf{c}^{\mathrm{T}} \wedge \forall i \in [\![m]\!]\ V_i^2 = (e^{v_i} f^{\rho_i})^4\Big\}$$

given parameters $(\mathbb{G}, P, n, Q, m, f)$ such that $f \in \mathbb{G}$ and $n, Q, m \in \mathbb{N}^*$, and $(\mathbf{g}, \mathbf{h}, \pi_{crs}) \in \mathbb{G}^{2n} \times \{0,1\}^*$. For fixed parameters n, Q and m, Section 6.1 shows that the protocol allows to prove the satisfiability of any polynomial in $\mathbb{Z}[X_1, \ldots, X_\nu]$ of total degree δ and with μ monomials if $\nu\lfloor \log \delta \rfloor + (\delta-1)\mu \leq n$ and $1 + 2\nu\,(\lfloor \log \delta \rfloor - 1) + (\delta - 2)\mu + m \leq Q$. The additional term m in the number of constraints compared to the previous section is to ensure the consistency between the committed variables $\mathbf{v}$ and the ones in the inner product.

Bünz et al. [10] gave a protocol for a similar relation in $\mathbb{Z}_p$ instead of $\mathbb{Z}$ to argue for the satisfiability of arithmetic circuits over $\mathbb{Z}_p$ (without the bounds related to integer polynomials as it was not their target) that is inspired by the one of Bootle et al. [8]. The general idea of our protocol for this relation is similar to the two previous ones, but there are key differences that arise from the fact that $\mathbb{Z}$ is not a field. These differences are highlighted in the overview below

Building Blocks. The protocol builds mainly on the protocol on Figure 2, and on three auxiliary protocols: a protocol Π_{crs} to prove that the CRS is well-formed (as in Section 5.3), a protocol Π' to aggregate arguments of opening to integer commitments (see Section 5.2) and a protocol $\tilde{\Pi}$ to argue knowledge of an integer vector that opens to commitments in different bases (Section 5.4), i.e., a base-switching argument. These arguments may be in the random-oracle model with an oracle $\mathcal{H}$.

Main Insights. The main idea of the protocol is to reduce the verification of the Hadamard product and of the linear constraints to a single inner-product argument over the integers, and then invoke the protocol on Figure 2.

Starting with the product $\mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O = 0_{\mathbb{Z}^n}$, the idea to verify all n equations at once is to consider each entry as the coefficient of a n-variate polynomial of total degree 1 and evaluate the polynomial at a vector $\mathbf{y} \leftarrow_\$ [\![0; P-1]\!]^n$ chosen by the verifier by computing $\langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O, \mathbf{y} \rangle$. If the polynomial is non-zero, then the evaluation will be zero will only negligible probability (this is the analog to the Schwartz-Zippel lemma in integral rings), i.e., $\langle \mathbf{a}_L \circ \mathbf{a}_R, \mathbf{y} \rangle = \langle \mathbf{a}_O, \mathbf{y} \rangle$ with high probability. The reason we choose $\mathbf{y} \leftarrow_\$ [\![0; P-1]\!]^n$ instead of choosing $y \leftarrow_\$ [\![0; P-1]\!]$ and setting $\mathbf{y} \leftarrow \begin{bmatrix} 1 & y & \cdots & y^{n-1} \end{bmatrix}$ (as did Bootle et al. and Bünz et al., and in which case the Schwartz-Zippel argument would still apply) is that it makes the integers in $\langle \mathbf{a}_L \circ \mathbf{a}_R, \mathbf{y} \rangle$ only mP times larger than those in $\mathbf{a}_L \circ \mathbf{a}_R$ instead of P^m. Similarly, the Q equations $\mathbf{W}_L \mathbf{a}_L^{\mathrm{T}} + \mathbf{W}_R \mathbf{a}_R^{\mathrm{T}} + \mathbf{W}_O \mathbf{a}_O^{\mathrm{T}} = \mathbf{W}_V \mathbf{v}^{\mathrm{T}} + \mathbf{c}^{\mathrm{T}}$ are reduced to a single equation by multiplying on the left by a random vector $\mathbf{z} \leftarrow_\$ [\![0; P-1]\!]^Q$. Naturally, the prover must commit to the inputs $\mathbf{a}_L, \mathbf{a}_R$ and the outputs $\mathbf{a}_O$ in the witness before receiving the values $\mathbf{y}$ and $\mathbf{z}$ for the Schwartz-Zippel lemma to apply since the coefficients of the polynomial cannot depend on the random point at which it is later evaluated. The inputs are committed in a group element C_I with some bases $(\mathbf{g}, \mathbf{h})$ and the outputs in a group element C_O with bases $\mathbf{g}$.

Now the goal is to verify both $\langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O, \mathbf{y} \rangle = 0$ and $\mathbf{z}\mathbf{W}_L \mathbf{a}_L^{\mathrm{T}} + \mathbf{z}\mathbf{W}_R \mathbf{a}_R^{\mathrm{T}} + \mathbf{z}\mathbf{W}_O \mathbf{a}_O^{\mathrm{T}} - \mathbf{z}\mathbf{W}_V \mathbf{v}^{\mathrm{T}} + \mathbf{z}\mathbf{c}^{\mathrm{T}} = 0$ with a single inner-product. To do so, we introduce another variable X and construct two vectors of polynomials of small degree $l(X)$ and $r(X)$ in $\mathbb{Z}^n[X]$ of which the inner product is denoted $t(X) := \langle l(X), r(X) \rangle$. The coefficients of $l(X)$ and $r(X)$ depend on the two relations to be proved, and are designed so that some coefficients of $t(X)$ force these relations while being computable from public values, i.e., so that the verifier can prevent the prover from cheating. The explicit construction of $l(X)$ and $r(X)$ is given below. After the prover has committed to the coefficient of $t(X)$ *that the verifier cannot compute on his own*, this polynomial is be evaluated at a random point x chosen by the verifier. However, the prover cannot simply send $t(x)$ to the verifier as it contains information about the witness, and adding a random integer will not help as $\mathbb{Z}$ is infinite unlike $\mathbb{Z}_p$. This is the first main difference with the argument of Bünz et al. For this reason, the verifier cannot check that $t(x)$ is correctly computed with the committed coefficients of t, and the argument cannot be directly reduced to the full inner-product argument in Section 4.

Fortunately, the prover can instead argue that she knows the committed coefficients of t and the openings to the committed $\mathbf{v}$ with the aggregated argument Π', assuming that they are committed in the same base (e, f). After these arguments, we notice that the verifier is able to compute $e^{t(x)}$ at any x of her choice from the coefficients of t that she could compute on her own (and which force the relations of interest with $\mathbf{y}$ and $\mathbf{z}$) and the committed coefficients. The verifier can then choose x, send it to the prover, and then together proceed with an argument of knowledge of $l(x)$ and $r(x)$ of which the inner product is committed to with the base e using the protocol on Figure 2.

It now remains to define such polynomials $l(X)$ and $r(X)$ in $\mathbb{Z}^n[X]$. Consider first the relation with $\mathbf{z}$. The goal is to make use of $\mathbf{W}_V$, $\mathbf{V}$ and $\mathbf{c}$ which are public to force the equality with the rest that contains private parts by making sure that the coefficient of $t(X)$ will be computable from the public information. As the relation is an inner product itself, notice that $\mathbf{a}_L$ and $\mathbf{z}\mathbf{W}_L$ cannot be in the same polynomial, and similarly for $\mathbf{a}_R$ and $\mathbf{z}\mathbf{W}_R$ and $\mathbf{a}_O$ and $\mathbf{z}\mathbf{W}_O$. If the relation $\mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O = 0_{\mathbb{Z}^n}$ were already enforced (with $\langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O, \mathbf{y}\rangle = 0$), one could put $\mathbf{a}_L$ and $\mathbf{z}\mathbf{W}_R$ at the same degree in $l(X)$, put $\mathbf{a}_R$ and $\mathbf{z}\mathbf{W}_L$ at the same degree in $r(X)$, and make sure that the sum of those degrees is equal to the degree at which $\mathbf{a}_0$ is in one of the polynomials (say $l(X)$), and leverage the equality $\mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O = 0_{\mathbb{Z}^n}$ to eliminate parasite terms that are not publicly computable. With this reasoning, the minimal degree $\mathbf{a}_0$ can be is then 2, and the others at degree 1.

The relation $\langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O, \mathbf{y}\rangle = 0_{\mathbb{Z}^n}$ must still be proved, and unfortunately since one cannot invert integers modulo the unknown orders of the bases, this verification cannot be embedded in the same degrees as the relation with $\mathbf{z}$. It must then be shifted to higher degrees, and a different polynomial than that of Bünz et al. must be defined (this is the second main difference). The same reasoning as before leads to additional terms $\mathbf{a}_L X^3 - \mathbf{a}_O X^4$ in $l(X)$ and to additional terms $\mathbf{y}X^2 - \mathbf{y} \circ \mathbf{a}_R X^3$ in $r(X)$, and the term of degree 6 of $t(X)$ is then 0.

The polynomials $l(X)$ and $r(X)$ are then respectively defined as $(\mathbf{a}_L + \mathbf{z}\mathbf{W}_R)\,X + \mathbf{a}_O X^2 + \mathbf{a}_L X^3 - \mathbf{a}_O X^4$ and $-\mathbf{1}^n + \mathbf{z}\mathbf{W}_O + (\mathbf{a}_R + \mathbf{z}\mathbf{W}_L)\,X + \mathbf{y}X^2 + \mathbf{y} \circ \mathbf{a}_R X^3$. Note that for $t(X) := \langle l(X), r(X)\rangle = \sum_{i=1}^{7} t_i X^i$ with

$$\begin{aligned}
t_2 &= \langle \mathbf{a}_L, \mathbf{z}\mathbf{W}_L\rangle + \langle \mathbf{z}\mathbf{W}_R, \mathbf{a}_R\rangle + \langle \mathbf{a}_O, \mathbf{z}\mathbf{W}_O\rangle + \langle \mathbf{z}\mathbf{W}_R, \mathbf{z}\mathbf{W}_L\rangle \\
&\quad - \langle \mathbf{a}_O, \mathbf{1}^n\rangle + \langle \mathbf{a}_L, \mathbf{a}_R\rangle \\
&= \langle \mathbf{a}_L, \mathbf{z}\mathbf{W}_L\rangle + \langle \mathbf{z}\mathbf{W}_R, \mathbf{a}_R\rangle + \langle \mathbf{a}_O, \mathbf{z}\mathbf{W}_O\rangle + \underbrace{\langle \mathbf{z}\mathbf{W}_R, \mathbf{z}\mathbf{W}_L\rangle}_{\delta(\mathbf{z})} \\
&= \langle \mathbf{z}\mathbf{W}_V, \mathbf{v}\rangle + \delta(\mathbf{z}) \quad \text{and} \\
t_6 &= \langle \mathbf{a}_L, \mathbf{y} \circ \mathbf{a}_R\rangle - \langle \mathbf{a}_O, \mathbf{y}\rangle = \langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_O, \mathbf{y}\rangle = 0.
\end{aligned}$$

Since $\mathbf{c}$ is public and the verifier can compute $\delta(\mathbf{z})$ on his own, the verifier can compute e^{t_2} from the commitments $\mathbf{V}$.

A last hurdle arises from the fact the verifier must be guaranteed that the vector $\mathbf{a}_R$ in the term $\mathbf{y} \circ \mathbf{a}_R X^3$ of $r(X)$ is really the same as the input vector to which the prover committed to at the beginning of the protocol. In $\mathbb{Z}_p$, one can easily do so by simply interpreting a commitment of the form $\mathbf{h}^{\mathbf{a}_R}$ as a commitment to $\mathbf{y} \circ \mathbf{a}_R$ in the basis $\left[h_1^{y_1^{-1}} \cdots h_n^{y_n^{-1}}\right]$. Nonetheless, the order of the group elements h_i are unknown in the present case (this is the third major difference with the proofs in groups of public order). That is why the prover must commit to $\mathbf{a}_L$ and $\mathbf{y} \circ \mathbf{a}_R$ in a new group element C_I' after receiving $\mathbf{y}$ and argue that $(\mathbf{a}_L, \mathbf{a}_R)$ opens to both C_I and C_I' respectively in the bases $(\mathbf{g}, \mathbf{h})$

and $(\mathbf{g}, \mathbf{h}')$ with $\mathbf{h}' = \left[h_1^{y_1} \cdots h_n^{y_n}\right]$. To do so, the prover and the verifier run the base-switching argument $\tilde{\Pi}$.

Protocol Algorithms. The protocol is denoted Π. The parameter-generation algorithm and the CRS generator are as in Section 4.1. The algorithms of the prover and the verifier are given on Figure 3. On that figure, $\mathbf{W}$ denotes the matrix $\left[\mathbf{W}_L\ \mathbf{W}_R\ \mathbf{W}_O\ \mathbf{W}_V\right]$. The values ℓ', $\tilde{\ell}$ and ℓ_2 are given in Section 6.2.

Prover-Communication Complexity. To estimate the communication complexity of the entire protocol from Figure 3, we estimate the complexity of each of its sub-protocol. To do so, one must assess the bit length of the integer witnesses given to each sub-protocol of the protocol on Figure 3, and this requires to give upper bounds on the polynomial evaluations in the protocol.

Heights of $l(X)$ and $r(X)$. Since $l(X) = (\mathbf{a}_L + \mathbf{z}\mathbf{W}_R) X + \mathbf{a}_O X^2 + \mathbf{a}_L X^3 - \mathbf{a}_O X^4 \in \mathbb{Z}^n[X]$, the height (i.e., the maximum of the absolute values of the coefficients) of the polynomials in $l(X)$ is at most $2^\ell - 1 + Q \left\| W \right\|_\infty (P-1)$ (recall that $\mathbf{W} = \left[\mathbf{W}_L\ \mathbf{W}_R\ \mathbf{W}_O\ \mathbf{W}_V\right]$). Concerning $r(X) = -\mathbf{1}^n + \mathbf{z}\mathbf{W}_O + (\mathbf{a}_R + \mathbf{z}\mathbf{W}_L) X + \mathbf{y}X^2 + \mathbf{y} \circ \mathbf{a}_R X^3$, the height of the polynomials in $r(X)$ is at most

$$\max\left(2^\ell - 1 + Q \left\| W \right\|_\infty (P-1), (P-1)(2^\ell - 1)\right) \leq P\left(2^\ell - 1 + Q \left\| W \right\|_\infty\right).$$

Prover Complexity in Π'. The height of $t(X) = \langle l(X), r(X) \rangle$ is at most

$$7nP\left(2^\ell - 1 + Q \left\| W \right\|_\infty P\right)\left(2^\ell - 1 + Q \left\| W \right\|_\infty\right).$$

The bit length of the height of $t(X)$ is thus at most

$$\begin{aligned} & 3 + \lfloor \log n \rfloor + \lfloor \log P \rfloor + 2 + \max\left(\ell, \lfloor \log(Q \left\| W \right\|_\infty) \rfloor + 1\right) + 1 \\ & \quad + \max\left(\ell, \lfloor \log P \rfloor + \lfloor \log(Q \left\| W \right\|_\infty) \rfloor + 2\right) + 1 \\ = {} & 7 + \lfloor \log n \rfloor + \lfloor \log P \rfloor + \max\left(\ell, \lfloor \log(Q \left\| W \right\|_\infty) \rfloor + 1\right) \\ & \quad + \max\left(\ell, \lfloor \log P \rfloor + \lfloor \log(Q \left\| W \right\|_\infty) \rfloor + 2\right) \\ \leq {} & 10 + \lfloor \log n \rfloor + 2\left(\lfloor \log P \rfloor + \ell + \lfloor \log(Q \left\| W \right\|_\infty) \rfloor\right). \end{aligned}$$

Therefore, the maximum bit length of the witness in this call is at most

$$\ell' \leftarrow \max\left(b_\mathbb{G} + \lambda + 4, 10 + \lfloor \log n \rfloor + 2\left(\lfloor \log P \rfloor + \ell + \lfloor \log(Q \left\| W \right\|_\infty) \rfloor\right)\right).$$

Π' is used to aggregate $m+5$ arguments, so the bit-communication complexity of Π' is of order $O\left(\ell' + \log(m+5) + \log P + \log(n) b_\mathbb{G}\right)$, i.e.,

$$O\left(\ell + \log(n) b_\mathbb{G} + \log Q + \log(m+5) + \log \left\| W \right\|_\infty\right).$$

Prover Complexity in $\tilde{\Pi}$. The bit length of the witness in this call is at most $\tilde{\ell} \leftarrow \max\left(b_\mathbb{G} + \lambda + 4, \ell\right)$. As $\tilde{\Pi}$ is used to argue about two bases, the bit complexity of $\tilde{\Pi}$ is of order $O\left(\tilde{\ell} + \log P + \log(n) b_\mathbb{G}\right)$, i.e., $O\left(\ell + \log(n) b_\mathbb{G}\right)$.

$\mathcal{P}\ (f, \mathbf{g}, \mathbf{h}, e, \pi_{crs}, \mathbf{W}, \mathbf{V}, \mathbf{c}, \ell; \mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O, \mathbf{v}, \rho)$ $\mathcal{V}$

$$\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \wedge \mathbf{W}_L \mathbf{a}_L^{\mathrm{T}} + \mathbf{W}_R \mathbf{a}_R^{\mathrm{T}} + \mathbf{W}_O \mathbf{a}_O^{\mathrm{T}} = \mathbf{W}_V \mathbf{v}^{\mathrm{T}} + \mathbf{c}^{\mathrm{T}} \wedge \forall i \in [\![m]\!]\ V_i^2 = (e^{v_i} f^{\rho_i})^4$$

$$\left\|\begin{bmatrix}\mathbf{a}_L & \mathbf{a}_R & \mathbf{a}_O & \mathbf{v} & \rho\end{bmatrix}\right\|_\infty < 2^\ell$$

return $\perp$ if $FS.\Pi_{crs}^{\mathcal{H}}.\mathsf{Vf}\left((\mathbb{G}, P, f, 2n+1), (\mathbf{g}, \mathbf{h}, e), \pi_{crs}\right) = 0$
$\rho_I, \rho_O \leftarrow_\$ \left[\!\left[0; 2^{b_\mathbb{G}+\lambda+3}\right]\!\right]$
$C_I \leftarrow (\mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} f^{\rho_I})^2$
$C_O \leftarrow (\mathbf{g}^{\mathbf{a}_O} f^{\rho_O})^2$

$$\xrightarrow{C_I, C_O}$$

$\mathbf{y} \leftarrow_\$ [\![0; P-1]\!]^n$
$\mathbf{z} \leftarrow_\$ [\![0; P-1]\!]^Q$

$$\xleftarrow{\mathbf{y},\mathbf{z}}$$

$l(X) \leftarrow (\mathbf{a}_L + \mathbf{z}\mathbf{W}_R)\, X + \mathbf{a}_O X^2 + \mathbf{a}_L X^3 - \mathbf{a}_O X^4$
$r(X) \leftarrow -\mathbf{1}^n + \mathbf{z}\mathbf{W}_O + (\mathbf{a}_R + \mathbf{z}\mathbf{W}_L)\, X + \mathbf{y}X^2 + \mathbf{y} \circ \mathbf{a}_R X^3$
$l(X), r(X) \in \mathbb{Z}^n[X]$ and $\mathbf{1}^n := \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix} \in \mathbb{Z}^n$
$t(X) \leftarrow \langle l(X), r(X) \rangle = \sum_{i=1}^{7} t_i X^i$
$t(X) \in \mathbb{Z}[X]$ and $t_6 = 0$
$\forall i \in [\![7]\!] \setminus \{2, 6\},\ s_i \leftarrow_\$ \left[\!\left[0; 2^{b_\mathbb{G}+\lambda+3}\right]\!\right], T_i \leftarrow \left(e^{t_i} f^{s_i}\right)^2$
Commit to the non-zero coefficients of $t(X)$ except for t_2
$\rho_I' \leftarrow_\$ \left[\!\left[0; 2^{b_\mathbb{G}+\lambda+3}\right]\!\right],\ C_I' \leftarrow \left(\mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{y}\circ\mathbf{a}_R} f^{\rho_I'}\right)^2$
Commit to the inputs in $(\mathbf{g}, \mathbf{h}')$ with $\mathbf{h}' := \begin{bmatrix}h_1^{y_1} & h_2^{y_2} & \cdots & h_n^{y_n}\end{bmatrix}$

$$\xrightarrow{\mathbf{T} := \begin{bmatrix}T_1 & T_3 & T_4 & T_5 & T_7\end{bmatrix}, C_I'}$$

$$\xleftrightarrow{(\tau', b') \leftarrow \langle \Pi'.\mathsf{Prove}(m+5, f, e, \mathbf{T}, \mathbf{V}, \ell'; \mathbf{t}, \mathbf{s}) \rightleftharpoons \Pi'.\mathsf{Vf}(m+5, f, e, \mathbf{T}, \mathbf{V}, \ell') \rangle}$$

Argue knowledge of representations of $(T_i)_{i\neq 2,6}$ and $(V_j)_{j=1}^m$ in (e, f)

if $b' = 0$ return $\perp$

$$\xleftrightarrow{\left(\tilde{\tau}, \tilde{b}\right) \leftarrow \left\langle \tilde{\Pi}.\mathsf{Prove}\left(2, f, (\mathbf{g},\mathbf{h}), (\mathbf{g},\mathbf{h}'), C_I, C_I', \tilde{\ell}; \mathbf{a}_L, \mathbf{a}_R, \rho_I, \rho_I'\right) \rightleftharpoons \tilde{\Pi}.\mathsf{Vf}\left(2, f, (\mathbf{g},\mathbf{h}), (\mathbf{g},\mathbf{h}'), C_I, C_I', \tilde{\ell}\right) \right\rangle}$$

Argue that C_I and C_I' are commitments to the same inputs in $(\mathbf{g}, \mathbf{h})$ and $(\mathbf{g}, \mathbf{h}')$

if $\tilde{b} = 0$ return $\perp$

$x \leftarrow_\$ [\![0; P-1]\!]$

$$\xleftarrow{x}$$

$\mathbf{l}, \mathbf{r} \leftarrow l(x), r(x) \in \mathbb{Z}^n$
$\sigma \leftarrow \rho_I x + \rho_I' x^3 + \rho_O \left(x^2 - x^4\right) + s_1 x + \langle \mathbf{z}\mathbf{W}_V, \rho \rangle\, x^2$
$+ \sum_{3 \leq i \neq 6 \leq 7} s_i x^i$ // $\rho = \begin{bmatrix}\rho_1 & \cdots & \rho_m\end{bmatrix}$

$$W_L, W_R, W_O \leftarrow \mathbf{h}^{\mathbf{z}\mathbf{W}_L}, \mathbf{g}^{\mathbf{z}\mathbf{W}_R}, \mathbf{h}^{\mathbf{z}\mathbf{W}_O}$$

$$C_{\mathbf{l},\mathbf{r}} \leftarrow C_I^x C_I'^{x^3} C_O^{(x^2 - x^4)} \left(\mathbf{h}^{-\mathbf{1}^n} \mathbf{h}'^{x^2} W_L^x W_R^x W_O\right)^2$$

$C_{\mathbf{l},\mathbf{r}}$ is a commitment to $\mathbf{l}$ and $\mathbf{r}$ in $(\mathbf{g}, \mathbf{h})$

$$C = C(x) \leftarrow C_{\mathbf{l},\mathbf{r}} T_1^x \left(e^{2(\langle \mathbf{z}, \mathbf{c} \rangle + \delta(\mathbf{z}))} \mathbf{V}^{\mathbf{z}\mathbf{W}_V}\right)^{x^2} \prod_{3 \leq i \neq 6 \leq 7} T_i^{x^i}$$

$$C = \left(\mathbf{g}^{\mathbf{l}} \mathbf{h}^{\mathbf{r}} e^{\langle \mathbf{l}, \mathbf{r} \rangle} f^\sigma\right)^2$$

run the protocol on Figure 2 on input $(1, n, f, \mathbf{g}, \mathbf{h}, e, C, \ell_2; \mathbf{l}, \mathbf{r}, \sigma)$

Fig. 3. Succinct Argument of Diophantine-Equation Satisfiability.

Prover Complexity in the Protocol on Figure 2. The largest absolute value of the components in $\mathbf{l} = l(x)$ and $\mathbf{r} = r(x)$ is at most $P^5\left(2^\ell - 1 + Q\|W\|_\infty\right)$. Besides, the height of the polynomial

$$\sigma(X) := \rho_I X + \rho_I' X^3 + \rho_O\left(X^2 - X^4\right) + s_1 X + \langle \mathbf{z}\mathbf{W}_V, \rho\rangle X^2 + \sum_{3\leq i\neq 6\leq 7} s_i x^i$$

is at most $2^{b_{\mathbb{G}}+\lambda+3} + mQ\|W\|_\infty P(2^\ell - 1)$ (given by the term of degree 2). The absolute value of randomness σ is thus at most $P^8\left(2^{b_{\mathbb{G}}+\lambda+3} + mQ\|W\|_\infty P\left(2^\ell - 1\right)\right)$. Therefore, the bit length of the integer witnesses (in absolute value) in the execution of the protocol on Figure 2 is at most

$$\ell_2 \leftarrow 5 + \lfloor \log P\rfloor + \max\left(b_{\mathbb{G}} + \lambda + 4, \ell + \lfloor \log\left(mQ\|W\|_\infty P\right)\rfloor + 1\right).$$

In case $m = 0$, it is only $b_{\mathbb{G}} + \lambda + 4$. The bit complexity of the call to this protocol is then of order

$$O\left(\ell + \log(n) b_{\mathbb{G}} + \log Q + \log m + \log\|W\|_\infty\right).$$

(The term $\log m$ vanishes in case $m = 0$.)

Overall Prover-Communication Complexity. Based on the previous estimations, the prover sends $O\left(\ell + \log(n) b_{\mathbb{G}} + \log Q + \log m + \log\|W\|_\infty\right)$ bits during the protocol (the term $\log m$ disappears in case $m = 0$). Therefore, for a polynomial in $\mathbb{Z}[X_1, \ldots, X_\nu]$ of total degree δ, with μ monomials and with coefficients less than 2^H in absolute value, assuming that $\nu\lfloor\log\delta\rfloor + (\delta - 1)\mu \leq n$ and $1 + 2\nu\left(\lfloor\log\delta\rfloor - 1\right) + (\delta - 2)\mu + m \leq Q$, the communication complexity of the protocol is of order

$$O\left(\ell + \log\left(\delta\binom{\nu+\delta}{\delta}\right) b_{\mathbb{G}} + H\right) = O\left(\ell + \min(\nu,\delta)\log\left(\nu+\delta\right) b_{\mathbb{G}} + H\right).$$

The term $H = \lfloor\log\|W\|_\infty\rfloor + 1$ comes from the fact that the procedure gives linear constraints determined by the coefficients of the polynomial.

Verification Effiency. Similarly to Section 4.1, the verifications of Π', $\tilde{\Pi}$ and the protocol on Figure 2 can each be done via single multi-exponentiations, with exponents of at most $O\left(\ell + b_{\mathbb{G}} + \log(n)\log(P) + \log Q + \log m + \log\|W\|_\infty\right)$ bits. For a polynomial in $\mathbb{Z}[X_1, \ldots, X_\nu]$ of total degree δ, with μ monomials and with coefficients less than 2^H in absolute value, that is $O\left(\ell + b_{\mathbb{G}} + \min(\nu,\delta)\log\left(\nu+\delta\right)\log P + H\right)$ bits.

6.3 Completeness and Security

This section formally states the properties achieved by the protocol.

Theorem 6.1 (Completeness). Π *is complete if* Π_{crs}, Π' *and* $\tilde{\Pi}$ *are.*

Proof. The completeness of Π immediately follows from its definition and from the completeness of Π_{crs}, Π' and $\tilde{\Pi}$, and the completeness of the protocol on Figure 2.

Theorem 6.2 (Honest-Verifier Zero-Knowledge Property). *If* Π' *and* $\tilde{\Pi}$ *are respectively* $(T_{\Pi'}, T_{\Pi'.\mathsf{Sim}}, \varepsilon_{\Pi'})$ *and* $\left(T_{\tilde{\Pi}}, T_{\tilde{\Pi}.\mathsf{Sim}}, \varepsilon_{\tilde{\Pi}}\right)$*-honest-verifier zero-knowledge, and if* $FS.\Pi^{\mathcal{H}}_{crs}$ *is is* $\left(T_{\Pi^{\mathcal{H}}_{crs}}, q_{\mathcal{H}}, \varepsilon^{\mathrm{snd}}_{\Pi^{\mathcal{H}}_{crs}}\right)$*-sound, then* Π *is* $\Big(T, O(b_{\mathbb{G}}) + T_{\Pi'.\mathsf{Sim}} + T_{\tilde{\Pi}.\mathsf{Sim}}$ $+T_{2.\mathsf{Sim}}, \varepsilon^{\mathrm{zk}}_{\Pi^{\mathcal{H}}_{crs}} + \varepsilon_{\Pi'} + \varepsilon_{\tilde{\Pi}} + \varepsilon_2\Big)$*- honest-verifier zero-knowledge for* $T \leq \min(T_{\Pi^{\mathcal{H}}_{crs}}, T_{\Pi'}, T_{\tilde{\Pi}}, T_2)$*, where* $(T_2, T_{2.\mathsf{Sim}}, \varepsilon_2)$ *denote the bounds from Theorem 4.2.*

Proof. It suffices to define a simulator which, instead of computing the commitments $C_I, C_O, C'_I, T_1, T_3, T_4, T_5, T_7$ in the protocol, computes elements as f^{α} for $\alpha \leftarrow_{\$} \left[\!\left[0; 2^{b_{\mathbb{G}}+\lambda+3}\right]\!\right]$. These are then at a statistical distance of at most $2^{-\lambda}$ from the values computed in a real protocol execution, unless the CRS is ill-formed (which occurs with probability at most $\varepsilon^{\mathrm{zk}}_{\Pi^{\mathcal{H}}_{crs}}$). The protocol also runs the simulator of Π', $\tilde{\Pi}$ and the simulator of the protocol on Figure 2.

Theorem 6.3 (Extractability). *If* $FS.\Pi_{crs}$ *is* $\left(T_{\tilde{\Pi}_{crs}.\mathsf{Sim}}, q_{\Pi_{crs}}, \varepsilon^{\mathrm{zk}}_{\tilde{\Pi}}\right)$*-statistically honest-verifier zero-knowledge,* Π' *is* $\left(T_{\Pi',\mathcal{A}}, T_{\Pi',\mathsf{Prove}^*}, T_{\Pi'.\mathcal{E}}, q_{\Pi'}, \varepsilon^{\mathrm{ext}}_{\Pi'}\right)$*-extractable and* $\tilde{\Pi}$ *is* $\left(T_{\tilde{\Pi}}, q_{\tilde{\Pi}}, \varepsilon^{\mathrm{snd}}_{\tilde{\Pi}}\right)$*-sound, then under the* $(T^{\mathrm{strg}}, \varepsilon^{\mathrm{strg}})$*-strong-root assumption, the* $\left(T^{\mathrm{ord}}, \varepsilon^{\mathrm{ord}}\right)$*-small-order assumption, the low-dyadic-valuation assumption and the* μ*-assumption over* G*, then* Π *is* $(T_{\mathcal{A}}, T_{\mathsf{Prove}^*}, T_{\mathcal{E}}, q_{\mathcal{H}}, \varepsilon^{\mathrm{ext}}, \Sigma)$*-extractable for* $T_{\mathcal{A}}$ *and* T_{Prove^*} *such that* $T_{\mathcal{A}} + T_{\mathsf{Prove}^*} \leq \min\left(T_{\tilde{\Pi}.\mathsf{Sim}}, T_{\tilde{\Pi}}\right)$ *and* $T_{\mathcal{A}}$ *and* T_{Prove^*} *satisfy the bounds for* $T_{\Pi',\mathcal{A}}$ *and* $T_{\Pi',\mathsf{Prove}^*}$ *and for* $T_{2,\mathcal{A}}$ *and* T_{2,Prove^*}*,* $T_{\mathcal{E}}$ *explicited in the proof of the theorem,* $q_{\mathcal{H}} \leq \min\left(q_{\Pi_{crs}}, q_{\tilde{\Pi}}, q_{\Pi'}\right)$ *and* $\varepsilon \leq \varepsilon^{\mathrm{zk}}_{\tilde{\Pi}} + \varepsilon^{\mathrm{ext}}_{\Pi'} + \varepsilon^{\mathrm{snd}}_{\tilde{\Pi}} + \varepsilon_2$*, where* $T_{2,\mathcal{A}}$*,* T_{2,Prove^*} *and* ε_2 *denote the bounds from Theorem 4.6.*

Proof. Suppose that for fixed vectors $\mathbf{y}$ and $\mathbf{z}$ and nine pairwise-distinct challenges $x_1, \ldots, x_9$, one can obtain representations $\left(\mathbf{g}^{\mathbf{l}}\mathbf{h}^{\mathbf{r}}e^{\langle \mathbf{l},\mathbf{r}\rangle}f^{\sigma}\right)^4$ of $C^2(x_j)$ for $j \in [\![8]\!]$ ($\mathbf{l}$, $\mathbf{r}$ and σ depend on x_j). Denoting by $\mathbf{X}$ the Vandermonde matrix of $x_1, \ldots, x_8$, by $\operatorname{adj}(X)$ its adjugate matrix and by $\mathbf{e}_j$ (for $j \in [\![8]\!]$) the canonical row vector with 1 at position j and 0 elsewhere, the linear equations $\mathbf{X}\nu^{\mathrm{T}} = \det(\mathbf{X})\mathbf{e}_j^{\mathrm{T}}$ with indeterminate $\nu \in \mathbb{Z}^8$ have unique solutions if $\det \mathbf{X} \neq 0$, and these solutions are $\operatorname{adj}(\mathbf{X})\mathbf{e}_j^{\mathrm{T}}$. Therefore, one can solve the equation $\mathbf{X}\nu^{\mathrm{T}} = \det(\mathbf{X})\mathbf{e}_2^{\mathrm{T}}$ and compute via linear combinations integer vectors $\mathbf{a}_L$ and $\mathbf{a}_R$ and an integer ρ_I such that $C_I^{2\det \mathbf{X}} = \left(\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R}e^{\langle \mathbf{a}_L, \mathbf{a}_R\rangle}f^{\rho_I}\right)^4$ since

$$C^2(x) = \left(C_I^x {C'_I}^{x^3} C_O^{x^2-x^4}\right)^2 \left(\mathbf{h}^{-1^n+\mathbf{y}x^2}\mathbf{h}^{x\mathbf{z}\mathbf{W}_L}\mathbf{g}^{x\mathbf{z}\mathbf{W}_R}\mathbf{h}^{\mathbf{z}\mathbf{W}_O}\right)^4 \left(T_1^x \left(e^{2(\langle \mathbf{z},\mathbf{c}\rangle+\delta(\mathbf{z}))}\mathbf{V}^{\mathbf{z}\mathbf{W}_V}\right)^{x^2} \prod_{3\leq i\neq 6\leq 7} T_i^{x^i}\right)^2 \tag{10}$$

for all $x \in \{x_1, \ldots, x_9\}$. Likewise, by considering the equation $\mathbf{X}\nu^{\mathrm{T}} = \det(\mathbf{X})\mathbf{e}_4^{\mathrm{T}}$, one can compute integer vector $\mathbf{a}'_L$ and $\mathbf{a}'_R$ and an integer ρ'_I such that ${C'_I}^{2\det \mathbf{X}} =$

$\left(\mathbf{g}^{\mathbf{a}'_L}\mathbf{h}^{\mathbf{a}'_R} e^{\langle \mathbf{a}'_L, \mathbf{a}'_R \rangle} f^{\rho'_I}\right)^4$. Moreover, let $\mathbf{e}$ be the row vector with 1 at position 3, -1 at position 5 and 0 elsewhere. The equation $\mathbf{X}\nu^{\mathrm{T}} = \det(\mathbf{X})\mathbf{e}^{\mathrm{T}}$ has a unique solution if $\det(\mathbf{X}) \neq 0$, and it is $\operatorname{adj}(\mathbf{X})\mathbf{e}^{\mathrm{T}}$. It follows that one can compute integer vectors $\mathbf{a}_{O,L}$ and $\mathbf{a}_{O,R}$ and an integer ρ_O such that $C_O^{2\det\mathbf{X}} = \left(\mathbf{g}^{\mathbf{a}_{O,L}}\mathbf{h}^{\mathbf{a}_{O,R}} e^{\langle \mathbf{a}_{O,L}, \mathbf{a}_{O,R} \rangle} f^{\rho_O}\right)^4$.

Besides, assume to be given integers t_j and s_j for $j \in [\![7]\!] \setminus \{2, 6\}$, and v_i and ρ_i for $i \in [\![m]\!]$ such that $T_j^2 = (e^{t_j} f^{s_j})^4$ and $V_i^2 = (e^{v_i} f^{\rho_i})^4$.

Lemma 4.4 shows that $2\det X$ divides all the integers in the representations of $C_I^{2\det\mathbf{X}}$, $C_I'^{2\det\mathbf{X}}$ and $C_O^{2\det\mathbf{X}}$ unless one can find non-trivial discrete-logarithm relations (the probability of this latter event is already accounted for in the bounds of Theorem 4.6). That is to say, one can obtain representations of C_I^2, $C_I'^2$ and C_O^2 (these integers obtained by dividing by $2\det X$ are further denoted as before). Inserting these representations in Equation 10 for any $x \in \{x_1, \dots, x_9\}$, it follows that

$$\begin{aligned}
\mathbf{l} &= (\mathbf{a}_L + \mathbf{z}\mathbf{W}_R)\, x + \mathbf{a}_{O,L} x^2 + \mathbf{a}'_L x^3 - \mathbf{a}_{O,L} x^4 \\
\mathbf{r} &= -\mathbf{1}^n + \mathbf{z}\mathbf{W}_O + (\mathbf{a}_R + \mathbf{z}\mathbf{W}_L)\, x + \mathbf{y} x^2 + \mathbf{a}'_R x^3 + \mathbf{a}_{O,R}\left(x^2 - x^4\right)
\end{aligned}$$

and that

$$\begin{aligned}
\langle \mathbf{l}, \mathbf{r} \rangle = \langle \mathbf{a}_L, \mathbf{a}_R \rangle\, x + \left\langle \mathbf{a}'_L, \mathbf{a}'_R \right\rangle x^3 + \langle \mathbf{a}_{O,L}, \mathbf{a}_{O,R} \rangle \left(x^2 - x^4\right) \\
+ s_1 x + (\langle \mathbf{z}\mathbf{W}_V, \mathbf{v} \rangle + \langle \mathbf{z}, \mathbf{c} \rangle + \delta(\mathbf{z}))\, x^2 + \sum_{3 \leq i \neq 6 \leq 7} t_i x^i,
\end{aligned}$$

unless one can obtain a non-trivial discrete-logarithm relation in $\langle f \rangle$. As the equation is satisfied for nine values of x although the polynomials are of degree at most 8 in x, then one can infer from the terms of degree 2, 6 and 8 that

$$\begin{aligned}
0 &= \langle \mathbf{a}_L, \mathbf{z}\mathbf{W}_L \rangle + \langle \mathbf{z}\mathbf{W}_R, \mathbf{a}_R \rangle + \langle \mathbf{a}_{O,L}, \mathbf{z}\mathbf{W}_O \rangle - \langle \mathbf{a}_{O,L}, \mathbf{1}^n \rangle \\
&\quad + \langle \mathbf{a}_L, \mathbf{a}_R \rangle - \langle \mathbf{z}\mathbf{W}_V, \mathbf{v} \rangle - \langle \mathbf{z}, \mathbf{c} \rangle - \langle \mathbf{a}_{O,L}, \mathbf{a}_{O,R} \rangle \\
0 &= \left\langle \mathbf{a}'_L, \mathbf{a}'_R \right\rangle - \langle \mathbf{a}_{O,L}, \mathbf{y} \rangle \quad \text{and} \\
0 &= \langle \mathbf{a}_{O,L}, \mathbf{a}_{O,R} \rangle .
\end{aligned}$$

Assuming an additional guarantee that $\mathbf{a}'_L = \mathbf{a}_L$ and $\mathbf{a}'_R = \mathbf{y} \circ \mathbf{a}_R$, these equations thus imply that

$$\begin{gathered}
\langle \mathbf{z}\mathbf{W}_L, \mathbf{a}_L \rangle + \langle \mathbf{z}\mathbf{W}_R, \mathbf{a}_R \rangle + \langle \mathbf{z}\mathbf{W}_O, \mathbf{a}_{O,L} \rangle = \langle \mathbf{z}\mathbf{W}_V, \mathbf{v} \rangle + \langle \mathbf{z}, \mathbf{c} \rangle \quad \text{and} \\
\langle \mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_{O,L}, \mathbf{y} \rangle = 0.
\end{gathered}$$

If these equalities are verified for m vectors $\mathbf{y}_1, \dots, \mathbf{y}_m \in \mathbb{Z}^m$ that are linearly independent over $\mathbb{Q}$ and for Q vectors $\mathbf{z}_1, \dots, \mathbf{z}_Q \in \mathbb{Z}^Q$ that are linearly independent over $\mathbb{Q}$, then $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_{O,L}$ and $\mathbf{W}_L\mathbf{a}_L^{\mathrm{T}} + \mathbf{W}_R\mathbf{a}_R^{\mathrm{T}} + \mathbf{W}_O\mathbf{a}_{O,L}^{\mathrm{T}} = \mathbf{W}_V\mathbf{v}^{\mathrm{T}} + \mathbf{c}^{\mathrm{T}}$, with v_i committed in V_i with randomness ρ_i for all $i \in [\![m]\!]$. In other words, $\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_{O,L}, \mathbf{v}, \rho$ is a valid witness.

The idea is now to gradually define an extractor for the entire protocol with a bottom-up approach starting from the extractor for the protocol on Figure 2 assuming vectors $\mathbf{y}, \mathbf{z}$ and an integer x to be fixed. The next extractor $\mathcal{E}_2$ builds on the previous extractor and can extract a witness given fixed vectors $\mathbf{y}$ and $\mathbf{z}$ (sent as the first message from the verifier) as inputs. The following oracle $\mathcal{E}_1$ takes a fixed vector $\mathbf{y}$ and builds on $\mathcal{E}_2$. The last extractor $\mathcal{E}$ takes no fixed values except for the protocol inputs builds then builds on $\mathcal{E}_1$.

The running time of each of these extractor has a success probability which of course depends on the success probability of the prover with the fixed inputs, and for instance, what $\mathcal{E}_1$ does is simply to generate a vector $\mathbf{z}$ that leads to a success, and then using a heavy-row argument, one can show that with probability $1/2$ over the choice of $\mathbf{z}$, the success of the prover decreases by a factor at most $1/2$ if $\mathbf{z}$ is fixed to the chosen value, and then $\mathcal{E}_1$ can run $\mathcal{E}_2$ with its fixed inputs and $\mathbf{z}$, and extract a witness in reasonable expected time. The idea for the other steps is the same.

Consider now an algorithm $\mathcal{E}_2$ that runs the prover from the computation step right after it receives $\mathbf{y}$ and $\mathbf{z}$, and suppose that for some fixed vectors $\mathbf{y}$ and $\mathbf{z}$, the success probability of the prover is at least $\eta > 0$. Algorithm $\mathcal{E}_2$ runs $\Pi'.\mathcal{E}$ on the prover to extract representation of $(T_i)_{i \neq 2,6}$ and $(V_j)_{j=1}^{m}$. It continues by executed protocol $\tilde{\Pi}$ with the prover. As $\tilde{\Pi}$ is assumed to be sound, the extractor is guaranteed that with probability at least $1 - \varepsilon_{\tilde{\Pi}}^{\mathrm{snd}}$, the inputs committed in C_I with bases $(\mathbf{g}, \mathbf{h})$ are the same as those committed in C_O with bases $(\mathbf{g}, \mathbf{h}')$ if the execution of $\tilde{\Pi}$ succeeds.

$\mathcal{E}_2$ generates $x \leftarrow_{\$} [\![0; P-1]\!]$, sends it to the prover and runs the protocol until the end. If the execution fails, $\mathcal{E}_2$ generates a fresh value x and proceeds as before. If the execution succeeds, then a heavy-row argument implies that with probability at least $1/2$ (over the choice of x), x is such that the prover succeeds in the rest of the protocol with probability at least $\eta/2$. Algorithm $\mathcal{E}_2$ then computes $C(x)$ as in the real protocol and runs the extractor of the protocol on Figure 2 (denote it $\Pi_2.\mathcal{E}$). If this latter does not return a value in at most twice the expected value of its running time with a prover that succeeds with probability at least $\eta/2$ (Markov's inequality shows that this event occurs with probability at most $1/2$), algorithm $\mathcal{E}_2$ generates a fresh value x and proceeds as before. If it does, $\mathcal{E}_2$ returns the same value.

Denote by H the event in which x is such that the verifier of is convinced with probability at least $\eta/2$ and by T the event in which $\Pi_2.\mathcal{E}$ returns a value in at most twice the expected value of its running time with a prover that succeeds with probability at least $\eta/2$. Algorithm $\mathcal{E}_2$ returns a value in the event $H \cap T$ and $\Pr[T \cap H] = \Pr[T|H]\Pr[H] \geq 1/4$, so $\mathcal{E}_2$ restarts from the generation of x an expected number of times at most 4. Furthermore, at each repetition, its running time is at most $T_{\mathsf{Prove}^*}/\eta + T_{\Pi_2.\mathcal{E}}(\eta/2)$, with $T_{\Pi_2.\mathcal{E}}(\eta/2)$ denoting the expected running time of $\Pi_2.\mathcal{E}$ with a prover that succeeds with probability at least $\eta/2$. Therefore, the expected running time given a prover that succeeds with probability at least η is at most $4\left(T_{\mathsf{Prove}^*}/\eta + 2T_{\Pi_2.\mathcal{E}}(\eta/2)\right)$.

$\mathcal{E}_2$ then repeats this process eight other times, with the restriction that the new challenge x_j (for $j = 2, \ldots, 9$) is distinct from the ones previously chosen ones, i.e., x_j has a uniform distribution over $[\![0; P-1]\!] \setminus \{x_1, \ldots, x_{j-1}\}$.

Note that in this case the prover succeeds with probability at least $\eta - (j-1)/P \geq \eta - 8/P =: \eta'$. Therefore, the total running time of $\mathcal{E}_2$ is at most $36\left(T_{\mathsf{Prove}^*}/\eta' + 2T_{\Pi_2.\mathcal{E}}(\eta'/2)\right)$.

Now, for a fixed vector $\mathbf{y}_1 \in \mathbb{Z}^n$, suppose that a prover convinces the verifier with probability at least $\theta > 0$ conditioned on the vector $\mathbf{y}$ in the first message from the verifeir being $\mathbf{y}_1$; and consider an algorithm $\mathcal{E}_1$ which has black-box access to a prover Prove^* and proceeds as follows. $\mathcal{E}_1$ starts by generating a vector $\mathbf{z}_1$. If $\mathbf{z}_1 = 0_{\mathbb{Z}^Q}$, then it generates new vectors and otherwise playing the role of the verifier, runs with the prover the entire protocol with $(\mathbf{y}_1, \mathbf{z}_1)$ as first message from the verifier. If the execution is unsuccessful, then $\mathcal{E}_1$ rewinds Prove^* to the beginning of the protocol and generates a new vector $\mathbf{z}_1$.

Note that conditioned on the event in which $\mathbf{z}_1$ is non-zero, Prove^* convinces the verifier with probability at least $\theta - P^{-Q}$. Moreover, conditioned on the event in which the first message $(\mathbf{y}_1, \mathbf{z}_1)$ leads to at least one successful execution, a heavy-row argument implies that with probability at least 1/2 (over the choice of $\mathbf{z}_1$), the first message $(\mathbf{y}_1, \mathbf{z}_1)$ is such that prover Prove^* convinces the verifier with probability at least $\left(\theta - P^{-Q}\right)/2$.

Algorithm $\mathcal{E}_1$ then runs $\mathcal{E}_2$ on Prove^* from the computation step right after the first message from the verifier is sent. If $\mathcal{E}_2$ returns a value in at most twice its expected running time with a prover that succeeds with probability at least $\eta := \left(\theta - P^{-Q}\right)/2$, then $\mathcal{E}_1$ returns that value and otherwise rewinds Prove^* to the beginning of the protocol and generates fresh vectors $\mathbf{y}_1$ and $\mathbf{z}_1$.

Similarly to the analysis of $\mathcal{E}_2$, the expected running of $\mathcal{E}_1$ is at most

$$4\left(\eta^{-1}\left(1 - P^{-Q}\right)^{-1} T_{\mathsf{Prove}^*} + 2T_{\mathcal{E}_2}(\eta/2)\right).$$

The term $\left(1 - P^{-Q}\right)^{-1}$ simply comes from the expected time necessary to generate a non-zero vector $\mathbf{z}_1$.

Now, for $j = 2, \ldots, Q$, algorithm $\mathcal{E}_2$ generates $\mathbf{z}_j \leftarrow_\$ [\![0; P-1]\!]^Q$. If $\mathbf{z}_1, \ldots, \mathbf{z}_j$ are linearly dependent over $\mathbb{Q}$, then $\mathcal{E}_2$ generates a new vector $\mathbf{z}_j$. Slinko [33, Corollary 2] proved that this event occurs with probability at most P^{-Q+j-1}. Consequently, conditioned on the event in which $\mathbf{z}_1, \ldots, \mathbf{z}_j$ are linearly independent, Prove^* convinces the verifier with probability at least $\theta - P^{-Q+j-1} \geq \theta - P^{-1}$. Algorithm $\mathcal{E}_2$ then proceeds as in the case $j = 1$ with $\theta - P^{-1}$ instead of $\theta - P^{-Q}$.

The total running time of $\mathcal{E}_1$ is thus at most

$$4Q\left(\eta^{-1}\left(1 - P^{-1}\right)^{-1} T_{\mathsf{Prove}^*} + 2T_{\mathcal{E}_2}(\eta/2)\right)$$

with $\eta := \left(\theta - P^{-1}\right)/2$. Note that if any two vectors $\mathbf{z}_i, \mathbf{z}_j$ for $i, j \in [\![Q]\!]$ lead to distinct witness returned by $\mathcal{E}_2$, then one obtains a non-trivial discrete-logarithm relation in $\langle f \rangle$.

The extractor $\mathcal{E}$ of the entire protocol can now be defined similarly to $\mathcal{E}_1$ to generate linearly independent vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{Z}^n$. Assuming that it has black-box access to a prover that succeeds with probability at least ε, its running time is at most

$$4n\left(\theta^{-1}\left(1-P^{-1}\right)^{-1}T_{\mathsf{Prove}^*}+2T_{\mathcal{E}_1}(\theta/2)\right)$$

with $\theta := \left(\varepsilon - P^{-1}\right)/2$. □

7 Applications

In this section, we apply the Diophantine-satisfiability argument from Section 6 to several computational problems. The methodology is always the same: encode the problem as a polynomial, apply the degree-reduction procedure from Section 6.1 and then run the protocol from Section 6.2.

7.1 Arguing Knowledge of RSA signatures

It is often useful for a user to prove that an organization has supplied him with a signature or a certificate without revealing it. A natural approach is to commit to the value of the signature and to prove its knowledge without revealing any information on it. If the user can additionally prove it in such a way that the signature cannot be linked to its issuance and multiple proofs cannot be linked to each other, then one can use the primitive in privacy-preserving applications. Camenisch and Stadler [12] presented a proof of knowledge of committed $x \in \mathbb{Z}_N^*$ such that $x^e = y \bmod N$, where (N, e) is an RSA public key. Their proof is in a group $\mathbb{G}$ of known order N (obtained for instance by finding a prime number p such that N divides $(p-1)$ and considering $\mathbb{G}$ as the subgroup of $\mathbb{Z}_p^*$ of order N). The main issue with this approach is that it requires at least that the prover knows N at the time the commitment scheme is set up. It also requires a new instance of the commitment scheme for each RSA moduli.

As mentioned above, Damgård and Fujisaki [15] proposed an efficient proof of knowledge of the contents of commitments and proofs of multiplicative relations over committed values. They stated that this scheme allows to prove in zero-knowledge the knowledge of an RSA signature since the verification equation can also be written as $x^e - \lambda N = y$ for some integer $\lambda < N^{e-1}$. This gives a proof with communication complexity with $\Omega(eb_{\mathbb{G}})$ bits which makes it usable only for very small e. One can use the elegant technique from [12] to improve this to $\Omega(\log(e)b_{\mathbb{G}})$ using the square-and-multiply algorithm in the exponent (see [12,13] for details). In the following, we show that our succinct Diophantine satisfiability argument can be used to exponentially reduce this communication complexity.

Let N be some *RSA* integer and let e be a public RSA exponent with $\mathbf{e} = (e_0, e_1, \ldots, e_{T-1}) \in \{0,1\}^T$ as binary representation, i.e.,

$$e = \sum_{i=0}^{T-1} e_i 2^i.$$

with $e_{T-1} = 1$. Suppose that one wants to prove the knowledge of some $x \in \mathbb{Z}_N^*$ such that $x^e = y \bmod N$ for some public value y (typically the hash value of a message). Suppose that the prover commits to x using our integer commitment scheme and wants to prove that the opening of V is an e-th root of the public y modulo N.

First define integers $(y_1, \ldots, y_{T-1})$ such that

$$y_i = \begin{cases} x \text{ if } e_i = 1 \\ 1 \text{ if } e_i = 0 \end{cases}$$

for $i \in \{0, \ldots, T-1\}$. Set then $z_{T-1} = x$ and define by induction

$$\begin{aligned} w_i &= z_i^2 \bmod N \\ \lambda_i &= \left\lfloor z_i^2/N \right\rfloor && \triangleright \quad w_i = z_i^2 - \lambda_i N && (11) \\ z_{i-1} &= y_i \cdot w_i \bmod N \\ \mu_i &= \lfloor (y_i \cdot w_i)/N \rfloor && \triangleright \quad z_{i-1} = y_i \cdot w_i - \mu_i N && (12) \end{aligned}$$

for i downward from $T-1$ to 0 such that if $x^e = y \bmod N$. Then, $z_{-1} = y$.

Reciprocally, if there exist five integer sequences $(w_0, \ldots, w_{T-1})$, $(y_0, \ldots, y_{T-1}, y_T)$, $(\lambda_0, \ldots, \lambda_{T-1})$, $(z_0, \ldots, z_{T-1})$, $(\mu_0, \ldots, \mu_{T-1})$ such that $w_0 y_0 - \mu_0 N = y$, Equations (11) and (12) hold for all $i \in \{0, \ldots, T-1\}$ and

$$\{y_i \colon e_i = 0, i \in \{0, \ldots, T-1\}\} = \{1\} \text{ and } \{y_i \colon e_i = 1, i \in \{0, \ldots, T\}\} = \{x\} \qquad (13)$$

where x, the unique element in the latter set, is committed in V, then $x^e = y \bmod N$.

One can now write down explicitly a Diophantine equation in such a way that the equation is satisfiable under the linear constraints (13) if and only if the value committed in V is an e-th root of the public y modulo N:

$$\sum_{i=0}^{T-1} \left(w_i - z_i^2 - \lambda_i N\right)^2 + \sum_{i=0}^{T-1} (z_i - y_i w_i - \mu_i N)^2 = 0$$

This Diophantine equation is directly suited for the reduction from Section 6.1 without introducing new variables. It indeed suffices to set

$$\begin{aligned} \mathbf{a}_L &\leftarrow \left[\mathbf{z} \left[\mathbf{w}\ 1\right] \boldsymbol{\lambda}\ \boldsymbol{\mu}\right] \\ \mathbf{a}_R &\leftarrow \left[\mathbf{z} \left[\mathbf{y}\ x\right] \mathbf{0}\ \mathbf{0}\right] \\ \mathbf{a}_O &\leftarrow \left[\mathbf{w} \left[y\ \mathbf{z}\right] \mathbf{0}\ \mathbf{0}\right] \in \mathbb{Z}^{4T+1}. \end{aligned}$$

Besides, there are $10T + 3$ linear equations that these vectors must satisfy ($2T$ for the terms in the sum of squares in the Diophantine equation, $4T$ for the zeros in $\mathbf{a}_R$ and $\mathbf{a}_0$, $2T$ for the consistency of $\mathbf{z}$, T for the consistency of $\mathbf{w}$, T for the consistency of $\mathbf{y}$ with V and 3 for the consistency of x and y).

Now, to estimate the bit complexity of the argument, note that each witness is upper-bounded by N and the maximum coefficient of the linear constraints is $O(N)$. According to the bounds in Section 6.2, the bit length of the argument is of order $O(\log(T)b_{\mathbb{G}} + \log N) = O(\log(\log(e))b_{\mathbb{G}} + \log N)$.

Finally, note that even if it is not necessary to commit to the value x, this technique can be used to argue knowledge of a modular e-th root modulo N with communication complexity $O(\log(\log(e))b_{\mathbb{G}})$ bits (in a group with $b_{\mathbb{G}} = \Omega(\log N)$). In particular for $e < 2^{O(\lambda/\log(\lambda))}$ this improves asymptotically the communication of the well-known protocol due to Guillou-Quisquater [26] which has communication complexity $O(\log(\lambda)/\log(e) \cdot \log(N))$.

7.2 Argument of Knowledge of (EC)DSA Signatures

The Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) [27] are standardized discrete-logarithm based efficient digital signature schemes. The underlying groups of public prime order are usually standardized and it may seem at first thought that one can use commitments in these groups (together with the classical zero-knowledge toolbox) in order to design a zero-knowledge argument of possession of an (EC)DSA signature. However, it turns out that this is not so easy since the verification equations of these signatures involve arithmetic modulo two different prime numbers. In this section, we show how one can use our Diophantine satisfiability argument and the approach we already used for RSA signatures to efficiently prove the knowledge of an (EC)DSA signature without revealing any further information.

DSA Signatures. The DSA signature scheme consists of the following procedures (given two parameters λ_1 and λ_2).

$\mathsf{KG}(\lambda_1, \lambda_2) \to (vk, sk)$: Generate a prime q of bit-length λ_2 uniformly at random and a prime p of bit-length λ_1 uniformly at random such that q divides $p-1$. Choose an integer $h \in \{2, \ldots, p-2\}$ uniformly at random and set $g = h^{(p-1)/q} \bmod p$ (in the rare case that $g = 1$, try again with a different h). Choose $x \in \mathbb{Z}_q$ uniformly at random, and compute the integer $y = g^x \bmod p$. The verification key is $vk \leftarrow (p, q, g, y, \mathcal{H})$ where $\mathcal{H}$ is a cryptographic hash function $\mathcal{H} \colon \{0,1\}^* \to \{0,1\}^{\lambda_2}$, while the signing key is $sk \leftarrow (p, q, g, x, \mathcal{H})$.

$\mathsf{Sign}(sk, m) \to \sigma$: Choose a secret $k \in \mathbb{Z}_q^*$ uniformly at random, then compute $g^k \bmod p$ and set $r = \left(g^k \bmod p\right) \bmod q$. Repeat the whole process until $r \neq 0$. Next, compute $s = k^{-1}(\mathcal{H}(m) + rx) \mod q$ and repeat the signing process from the beginning until $s \neq 0$. The signature of the message m is the pair $\sigma \leftarrow (r, s)$.

$\mathsf{Vf}(vk, m, \sigma) \to b \in \{0,1\}$: To verify a signature pair $\sigma = (r, s)$ of a message m for vk, first check that $0 < r, s < q$. Then, compute

- $w = s^{-1} \bmod q$
- $u_1 = \mathcal{H}(m) \cdot w \bmod q$
- $u_2 = r \cdot w \bmod q$

– $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$

Accept the signature (i.e., return 1) if and only $r = v$.

If a user wants to prove the knowledge of such a signature $\sigma = (r, s)$ for a public message m and a public verification key is $vk = (p, q, g, y, \mathcal{H})$, she has to prove the knowledge of a 6-tuple of integers $(r, w, \alpha, \beta, \mu_1, \mu_2)$ such that

$$\begin{aligned}
\alpha &= \mathcal{H}(m) \cdot w - \mu_1 \cdot q \\
\beta &= r \cdot w - \mu_2 \cdot q \\
\beta &\neq 0 \\
r &= \left(g^{\alpha} y^{\beta} \bmod p\right) \bmod q.
\end{aligned}$$

In order to prove the last equality, introduce the value $\rho = \left(g^{\alpha} y^{\beta} \bmod p\right)$ and μ_3 such that

$$r = \rho - \mu_3 \cdot q.$$

Write $g_i = g^{2^i} \bmod p$ and $y_i = y^{2^i} \bmod p$ for $i \in \{0, \ldots, \lambda_2 - 1\}$ and

$$\alpha = \sum_{i=0}^{\lambda_2 - 1} \alpha_i 2^i \text{ with } \alpha_i \in \{0, 1\} \text{ for } i \in \{0, \ldots, \lambda_2 - 1\}$$

$$\beta = \sum_{i=0}^{\lambda_2 - 1} \beta_i 2^i \text{ with } \alpha_i \in \{0, 1\} \text{ for } i \in \{0, \ldots, \lambda_2 - 1\}.$$

The values α_i and β_i for $i \in \{0, \ldots, \lambda_2 - 1\}$ have to be kept secret by the user. Starting from $\rho_{-1} = 1$, recursively construct

$$\begin{aligned}
\nu_i &= \rho_{i-1} \cdot (1 + \alpha_i \cdot (g_i - 1)) \bmod p & \triangleright\, \nu_i &= \rho_{i-1} \cdot (1 + \alpha_i \cdot (g_i - 1)) + \delta_i p \\
\rho_i &= \nu_i \cdot (1 + \beta_i \cdot (y_i - 1)) \bmod p & \triangleright\, \rho_i &= \nu_i \cdot (1 + \beta_i \cdot (y_i - 1)) + \varepsilon_i p
\end{aligned}$$

such that $\rho_j = g^{\sum_{i=0}^{j} \alpha_i 2^i} y^{\sum_{i=0}^{j} \beta_i 2^i} \bmod p$ for $j \in \{0, \ldots, \lambda_2 - 1\}$ and in particular $\rho_{\lambda_2} = \rho$. It thus leads to the Diophantine equation

$$\begin{aligned}
&\sum_{i=0}^{\lambda_2 - 1} \left(\alpha_i^2 - \alpha_i\right)^2 + \left(\beta_i^2 - \beta_i\right)^2 + (\nu_i - \rho_{i-1} \cdot (1 + \alpha_i \cdot (g_i - 1)) - \delta_i p)^2 \\
&\quad + \sum_{i=0}^{\lambda_2 - 1} (\rho_i - \nu_i \cdot (1 + \beta_i \cdot (y_i - 1)) - \varepsilon_i p)^2 + (r - \rho_{\lambda_2} + \mu_3 \cdot q)^2 \\
&\quad + (\alpha - \mathcal{H}(m) \cdot w + \mu_1 \cdot q)^2 + (\beta - r \cdot w + \mu_2 \cdot q)^2 \\
&\quad + \left(\alpha - \sum_{i=0}^{\lambda_2 - 1} \alpha_i 2^i\right)^2 + \left(\beta - \sum_{i=0}^{\lambda_2 - 1} \beta_i 2^i\right)^2 = 0,
\end{aligned}$$

with $(\alpha, \alpha_0, \ldots, \alpha_{\lambda_2})$, $(\beta, \beta_0, \ldots, \beta_{\lambda_2})$, $(\rho_0, \ldots, \rho_{\lambda_2})$, $(\nu_0, \ldots, \nu_{\lambda_2})$, $(\delta_0, \ldots, \delta_{\lambda_2})$, $(\epsilon_0, \ldots, \epsilon_{\lambda_2})$ and $(r, w, \mu_1, \mu_2, \mu_3)$ as unknowns.

Using the approach from Section 6.2, the bit length of the argument is of order $O\left(\log(\lambda_2) b_{\mathbb{G}} + \log(\lambda_1)\right)$. Note also that this argument could be combined with proofs of non-algebraic statements [13] to obtain proofs on committed messages.

ECDSA Signatures. For ECDSA signatures, the underlying group is an elliptic curve $E(\mathbb{Z}_p)$ defined over $\mathbb{Z}_p$. The group law involves arithmetic operations modulo p, but the group order is usually some prime number q (of bit size equal to the bit size of p). The main difference in an ECDSA signature $(r, s) \in \mathbb{Z}_p^2$ is that the verification equation checks that r is the abscissa of some point on the elliptic curve obtained by a double scalar multiplication with exponents derived from (r, s) and the hash value of the signed message. It is therefore possible to follow the same strategy as for DSA by adding more variables in order to handle the more intricate group law on the elliptic curve. More precisely, one need only add a constant number of variables per bit in the exponents and using the approach from Section 6.2, the bit length of the argument is of order $O\left(\log(\log p) b_{\mathbb{G}}\right)$.

7.3 Argument of Knowledge of List Permutation

Consider the classical problem of proving knowledge of openings $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ (together with the corresponding randomness) of two commitment vectors $(V_1, \ldots, V_n)$ and $(V_1', \ldots, V_n')$, and of a permutation $\pi \in \mathfrak{S}_n$ (the symmetric group of order n) such that $y_i = x_{\pi(i)}$ for all $i \in [\![n]\!]$.

To provide a succinct argument of such a statement, one can write down a simple Diophantine equation that is satisfiable (under additional linear constraints) if and only if the statement indeed holds. Denote $V_i = e^{x_i} f^{\rho_i}$ and $V_i' = e^{y_i} f^{\eta_i}$ for $i \in [\![n]\!]$. Note that if there exists such a permutation π, one can consider the associated permutation binary matrix $U = (u_{i,j}) \in \mathbb{Z}^{n \times n}$ defined by $u_{i,j} = 1$ if $j = \pi(i)$ and $u_{i,j} = 0$ otherwise. Such a matrix has exactly one "1" per row and per column (and is null at all other indices). Therefore, $x_i = y_j$ for integers $i, j \in [\![n]\!]$ if $u_{i,j} = 1$. In particular, for all $i \in [\![n]\!]$

$$\sum_{j=1}^{n} \left(u_{i,j}\left(x_i - y_j\right)\right)^2 = 0. \tag{14}$$

Conversely, if there exists a permutation matrix $\mathbf{U} = (u_{i,j}) \in \mathbb{Z}^{n \times n}$ such that Equation (14) holds for all $i \in [\![n]\!]$, then there exist a permutation $\pi \in \mathfrak{S}_n$, such that $y_i = x_{\pi(i)}$ for all $i \in [\![n]\!]$. Consider then the following Diophantine equation

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \left(u_{i,j}\left(x_i - y_j\right)\right)^2 + \left(u_{i,j}^2 - u_{i,j}\right)^2 = 0$$

with the $2n$ linear constraints:

$$\sum_{i=1}^{n} u_{i,j} = 1 \text{ for all } j \in [\![n]\!] \quad \sum_{j=1}^{n} u_{i,j} = 1 \text{ for all } i \in [\![n]\!]\,.$$

Note that these constraints could have been direclty embedded in the Diophantine equation.

Following the ideas in Section 6.1, by introducing variables $v_{i,j} \leftarrow u_{i,j}x_i$ for $i, j \in [\![n]\!]$, the previous equation is equivalent to the satisfiability of the equation

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \left(v_{i,j} - u_{i,j}y_j\right)^2 + \left(u_{i,j}^2 - u_{i,j}\right)^2 + \left(v_{i,j} - u_{i,j}x_i\right)^2 = 0.$$

This polynomial is now in a form which allows to immediately read a Hadamard product between variables and linear equations of which the satisfiability is equivalent to the satisfiability of the Diophantine equation. It suffices to set

$$\mathbf{a}_L \leftarrow \left[\hat{\mathbf{x}}\ \hat{\mathbf{y}}\ \mathbf{u}\right] \quad \mathbf{a}_R \leftarrow \left[\mathbf{u}\ \mathbf{u}\ \mathbf{u}\right] \quad \mathbf{a}_O \leftarrow \left[\mathbf{v}\ \mathbf{v}\ \mathbf{u}\right] \in \mathbb{Z}^{3n^2}.$$

where $\mathbf{u}$ is the vector of dimension n^2 obtained by concatenating the rows of $\mathbf{U}$, $\hat{\mathbf{x}}$ is the vector of dimension n^2 obtained by repeating n times each coordinate of $\mathbf{x}$ and $\hat{\mathbf{y}} = \left[\mathbf{y} \cdots \mathbf{y}\right]$ is a vector of dimension n^2.

Besides, there are $7n^2 + 2n$ linear equations that these vectors must satisfy and the vectors $\mathbf{x}$ and $\mathbf{y}$ are committed individually (i.e., $2n$ commitments). To estimate the bit complexity of the argument, note that if have $\|\mathbf{x}\|_\infty < 2^\ell$ (and thus $\|\mathbf{y}\|_\infty < 2^\ell$), then, according to the analysis in Section 6.2, the bit length of the argument is of order $O\left(\ell + \log\left(3n^2\right) b_{\mathbb{G}}\right) = O\left(\ell + \log(n) b_{\mathbb{G}}\right)$.

7.4 3-SAT Satisfiability Argument

3-SAT is the prototypical NP-complete problem of deciding the satisfiability of a Boolean formula in conjunctive normal form, with each clause limited to at most three literals. Consider two integers $m, n \geq 1$ and a set of clause $C_1, \ldots, C_m$ where each C_i is the disjunction of exactly three literals from the set of Boolean variables $x_1, \ldots, x_n$ (a literal is a variable x_i or its negation $\neg x_i$). The problem is decide whether there exists an assignment of $(x_1, \ldots, x_n) \in \{0, 1\}^n$ such that all clauses are satisfied. Write each clause C_i for $i \in [\![m]\!]$ as $C_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$, with $l_{i,1}$, $l_{i,2}$ and $l_{i,3}$ denoting the literals in C_i.

Such a Boolean 3-SAT formula can be readily turned into an equi-satisfiable Diophantine equation. Indeed, for $i \in [\![m]\!]$ and $j \in [\![3]\!]$ and $k \in [\![n]\!]$, define

$$\varepsilon_{3(i-1)+j,k} = \begin{cases} 1 \text{ if } l_{i,j} = x_k \\ -1 \text{ if } l_{i,j} = \neg x_k \\ 0 \text{ otherwise} \end{cases}$$

so that setting $\mathbf{l} := \left[l_{1,1}\ l_{1,2}\ l_{1,3} \cdots l_{m,1}\ l_{m,2}\ l_{m,3}\right]$, $\mathbf{x} := \left[x_1 \cdots x_n\right]$ and $\mathbf{E} := \left[\varepsilon_{i,k}\right]_{1 \leq i \leq 3m, 1 \leq k \leq n}$, the equality ${}^{\mathrm{T}}\mathbf{l} = \mathbf{A} \cdot \mathbf{x}^{\mathrm{T}}$ holds. The Diophantine equation must ensure that (1) $x_i \in \{0, 1\}$, i.e., $x_j\left(1 - x_j\right) = 0$ for $j \in [\![n]\!]$, and that (2) the clauses are all satisfied, i.e., $(1 - l_{i,1})(1 - l_{i,2})(1 - l_{i,3}) = 0$ for all $i \in [\![m]\!]$. The satisfiability of the 3-SAT instance is then equivalent to the satisfiability of the Diophantine equation

$$\sum_{j=1}^{n}\left(x_j - x_j^2\right)^2 + \sum_{i=1}^{m}\left(1 - l_{i,1} - l_{i,2} - l_{i,3} + l_{i,1}l_{i,2} + l_{i,1}l_{i,3} + l_{i,2}l_{i,3} - l_{i,1}l_{i,2}l_{i,3}\right)^2 = 0$$

Following the ideas in Section 6.1, by introducing variables $u_{i,1} \leftarrow l_{i,1}l_{i,2}$, $u_{i,2} \leftarrow l_{i,1}l_{i,3}$, $u_{i,3} \leftarrow l_{i,2}l_{i,3}$, $v_i \leftarrow u_{i,1}l_{i,3}$ for $i \in [\![m]\!]$, the previous equation is equivalent to the satisfiability of the equation

$$\sum_{j=1}^{n}\left(x_j - x_j^2\right)^2 + \sum_{i=1}^{m}\left(1 - l_{i,1} - l_{i,2} - l_{i,3} + u_{i,1} + u_{i,2} + u_{i,3} - v_i\right)^2$$
$$+ \sum_{i=1}^{m}\left((u_{i,1} - l_{i,1}l_{i,2})^2 + (u_{i,2} - l_{i,1}l_{i,3})^2 + (u_{i,3} - l_{i,2}l_{i,3})^2 + (v_i - u_{i,1}l_{i,3})\right)^2 = 0.$$

This polynomial is now in a form which allows to immediately read a Hadamard product between variables and linear equations of which the satisfiability is equivalent to the satisfiability of the Diophantine equation. It suffices to set

$$\begin{aligned}\mathbf{a}_L &\leftarrow \left[\mathbf{x}\ \mathbf{l}^{(1)}\ \mathbf{l}^{(1)}\ \mathbf{l}^{(2)}\ \mathbf{l}^{(3)}\right]\\ \mathbf{a}_R &\leftarrow \left[\mathbf{x}\ \mathbf{l}^{(2)}\ \mathbf{l}^{(3)}\ \mathbf{l}^{(3)}\ \mathbf{u}_1\right]\\ \mathbf{a}_O &\leftarrow \left[\mathbf{x}\ \mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3\ \mathbf{v}\right] \in \mathbb{Z}^{n+4m}.\end{aligned}$$

Besides, there are $2n + 9m$ linear equations that these vectors must satisfy ($3m$ from matrix $\mathbf{E}$, m for the final Diophantine equation and $2n + 5m$ for the consistencies in $\mathbf{a}_L$, $\mathbf{a}_R$ and $\mathbf{a}_0$).

Now, to estimate the bit complexity of the argument, note that since $\|\mathbf{x}\|_\infty \leq 1$ and $\|\mathbf{E}\|_\infty \leq 1$, all witnesses and all variables are upper-bounded by 1, and according to the bounds in Section 6.2, the bit length of the argument is of order $O(\log(n + 4m)b_{\mathbb{G}}) = O(\log(n + m)b_{\mathbb{G}})$.

7.5 Integer-Linear-Programming Satisfiability Argument

A (decisional) Integer-Linear-Programming (ILP) problem is a feasibility linear program in which the variables are integers: given two positive integers m and n, a matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$ and a vector $\mathbf{b} \in \mathbb{Z}^m$, the problem consists in deciding whether there exists a vector $\mathbf{x} \in \mathbb{N}^n$ such that $\mathbf{A}\mathbf{x}^{\mathrm{T}} \geq \mathbf{b}^{\mathrm{T}}$. This problem is a classical NP-complete problem and it models many real-life optimization problems, and the following shows how to succinctly argue knowledge of a solution to it using the techniques presented in Section 6.

In other to prove the positivity of an integer x, as previously done in the literature [9,14,28], one can rely on Lagrange's four-square theorem which states that every natural integer can be represented as the sum of four integer squares. Actually, as remarked by Groth [21], one can also rely on Legendre's three-square theorem which states that every natural number $x = 1 \bmod 4$ can be represented as the sum of three integer squares. Both theorems are effective and there exists efficient polynomial-time algorithms to find the decomposition as a sum of squares (see [9, 14, 21, 28] and references therein for details).

To argue the satisfiability of the ILP problem, it is thus equivalent to argue knowledge of vectors $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3 \in \mathbb{Z}^n$ and vectors $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3 \in \mathbb{Z}^m$ such that

$$4x_j + 1 - y_{1,j}^2 - y_{2,j}^2 - y_{3,j}^2 = 0 \quad \forall j \in [\![n]\!] \text{ and}$$
$$\sum_{j=1}^{n} 4a_{ij}x_j - 4b_i + 1 - z_{1,i}^2 - z_{2,i}^2 - z_{3,i}^2 = 0 \quad \forall i \in [\![m]\!] \,.$$

The satisfiability of the ILP problem is then equivalent to the satisfiability of the Diophantine equation

$$\sum_{j=1}^{n} \left(4x_j - y_{1,j}^2 - y_{2,j}^2 - y_{3,j}^2 + 1\right)^2 + \sum_{i=1}^{m} \left(\sum_{j=1}^{n} 4a_{ij}x_j - 4b_i - z_{1,i}^2 - z_{2,i}^2 - z_{3,i}^2 + 1\right)^2 = 0.$$

Following the ideas in Section 6.1, by introducing variables $u_{1,j} \leftarrow y_{1,j}^2$, $u_{2,j} \leftarrow y_{2,j}^2$, $u_{3,j} \leftarrow y_{3,j}^2$, $v_{1,i} \leftarrow z_{1,i}^2$, $v_{2,i} \leftarrow z_{2,i}^2$ and $v_{3,i} \leftarrow z_{3,i}^2$, the previous equation equation is equivalent to the satisfiability of the equation

$$\begin{aligned}&\sum_{j=1}^{n} \left(4x_j - u_{1,j} - u_{2,j} - u_{3,j} + 1\right)^2 + \sum_{i=1}^{m} \left(\sum_{j=1}^{n} 4a_{ij}x_j - 4b_i - v_{1,i} - v_{2,i} - v_{3,i} + 1\right)^2 \\ &+ \sum_{j=1}^{n} \left(u_{1,j} - y_{1,j}^2\right)^2 + \left(u_{2,j} - y_{2,j}^2\right)^2 + \left(u_{3,j} - y_{3,j}^2\right)^2 \\ &+ \sum_{i=1}^{m} \left(v_{1,i} - z_{1,i}^2\right)^2 + \left(v_{2,i} - z_{2,i}^2\right)^2 + \left(v_{3,i} - z_{3,i}^2\right)^2 = 0.\end{aligned}$$

This polynomial is now in a form which allows to immediately read a Hadamard product between variables and linear equations of which the satisfiability is equivalent to the satisfiability of the Diophantine equation. It suffices to set

$$\begin{aligned}\mathbf{a}_L &\leftarrow \left[\mathbf{x}\ \mathbf{y}_1\ \mathbf{y}_2\ \mathbf{y}_3\ \mathbf{z}_1\ \mathbf{z}_2\ \mathbf{z}_3\right] \\ \mathbf{a}_R &\leftarrow \left[\mathbf{0}\ \mathbf{y}_1\ \mathbf{y}_2\ \mathbf{y}_3\ \mathbf{z}_1\ \mathbf{z}_2\ \mathbf{z}_3\right] \\ \mathbf{a}_O &\leftarrow \left[\mathbf{0}\ \mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3\ \mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3\right] \in \mathbb{Z}^{4n+3m}.\end{aligned}$$

Besides, there are $4(n + m)$ linear equations that these vectors must satisfy.

Now, to estimate the bit complexity of the argument, note that if $\|\mathbf{x}\|_\infty < 2^\ell$, then $\left|\sum_{j=1}^{n} 4a_{ij}x_j - 4b_i + 1\right| < 4\left(n\,\|\mathbf{A}\|_\infty\,2^\ell + \|\mathbf{b}\|_\infty\right) + 1$ for all $i \in [\![m]\!]$. Therefore, according to the bounds in Section 6.2, the bit length of the argument is of order $O\left(\ell + \log(4n + 3m)b_{\mathbb{G}} + \log\|\mathbf{A}\|_\infty + \log\|\mathbf{b}\|_\infty\right)$.

Acknowledgements

This work was supported by the French ANR ALAMBIC Project (ANR-16-CE39-0006) and the EU H2020 Research and Innovation Program under Grant Agreement No. 786725 (OLYMPUS).

References

1. M. Artin. *Algebra*. Pearson, 2010.
2. R. M. Avanzi. The complexity of certain multi-exponentiation techniques in cryptography. *Journal of Cryptology*, 18(4):357–373, Sept. 2005.
3. E. H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comput.*, 22(103):565–578, 1968.
4. N. Bari and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 480–494. Springer, Heidelberg, May 1997.
5. S. Bayer and J. Groth. Zero-knowledge argument for polynomial evaluation with application to blacklists. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 646–663. Springer, Heidelberg, May 2013.
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.
7. I. Biehl, J. A. Buchmann, S. Hamdy, and A. Meyer. A signature scheme based on the intractability of computing roots. *Des. Codes Cryptogr.*, 25(3):223–236, 2002.
8. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.
9. F. Boudot. Efficient proofs that a committed number lies in an interval. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 431–444. Springer, Heidelberg, May 2000.
10. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.
11. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020.
12. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 410–424. Springer, Heidelberg, Aug. 1997.
13. M. Chase, C. Ganesh, and P. Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 499–530. Springer, Heidelberg, Aug. 2016.
14. G. Couteau, T. Peters, and D. Pointcheval. Removing the strong RSA assumption from arguments over the integers. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 321–350. Springer, Heidelberg, Apr. / May 2017.
15. I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Y. Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142. Springer, Heidelberg, Dec. 2002.
16. R. del Pino, V. Lyubashevsky, and G. Seiler. Short discrete log proofs for FHE and ring-LWE ciphertexts. In D. Lin and K. Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 344–373. Springer, Heidelberg, Apr. 2019.

17. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, Aug. 1987.
18. P.-A. Fouque and G. Poupard. On the security of RDSA. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 462–476. Springer, Heidelberg, May 2003.
19. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 16–30. Springer, Heidelberg, Aug. 1997.
20. J. v. z. Gathen and J. Gerhard. *Modern Computer Algebra.* Cambridge University Press, USA, 3rd edition, 2013.
21. J. Groth. Non-interactive zero-knowledge arguments for voting. In J. Ioannidis, A. Keromytis, and M. Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 467–482. Springer, Heidelberg, June 2005.
22. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, Dec. 2006.
23. J. Groth. Linear algebra with sub-linear zero-knowledge arguments. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 192–208. Springer, Heidelberg, Aug. 2009.
24. J. Groth and M. Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 253–280. Springer, Heidelberg, Apr. 2015.
25. J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006.
26. L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 123–128. Springer, Heidelberg, May 1988.
27. D. Johnson, A. Menezes, and S. A. Vanstone. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Sec.*, 1(1):36–63, 2001.
28. H. Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In C.-S. Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 398–415. Springer, Heidelberg, Nov. / Dec. 2003.
29. Y. V. Matiyasevich. Enumerable sets are diophantine. *Sov. Math., Dokl.*, 11:354–358, 1970.
30. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, Aug. 1992.
31. C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, Jan. 1991.
32. T. Skolem. *Diophantische Gleichungen.* Ergebnisse der Mathematik und ihrer Grenzgebiete. New York, Chelsea Pub. Co., 1950.
33. A. Slinko. A generalization of Komlos theorem on random matrices. *New Zealand J. Math.*, 30(1):81–86, 2001.

A Succinct Inner-Product Argument on Integers

This section gives further details about the integer-product argument system given in Section 4.

A.1 Verification via a Single Multi-Exponentiation

In this section, we tackle the verification of inner-product arguments via single multi-exponentiation in case the vector length is not a power of 2. To do so, we give an explicit expression of the bases at the last step of the recursion. Alternatively, we give an algorithm which allows to always reduce to the case of powers of 2 by inserting neutral elements at specific positions in the original vector.

Explicit Expression for the Final Bases. This section gives an explicit expression of the bases g and h at the last step of the recursion in terms of $\mathbf{g}$, $\mathbf{h} \in \mathbb{G}^n$ and the x_j challenges for $j \in [\![n']\!]$ (with $n' = \lceil \log n \rceil$) in case n is not a power of two. Recall from Lemma A.1 that the size of the vectors of group elements at the beginning of the ith recursion step is $\left\lceil n/2^{-i} \right\rceil$.

First consider the case of g. Notice that the g_i elements that are raised to the power x_1 in the expression of g are such that $1 \leq i \leq \lfloor n/2 \rfloor$, and the elements that are raised to the power x_2 are such that $1 \leq i \leq \lfloor \lceil n/2 \rceil /2 \rfloor$ or $\lfloor n/2 \rfloor + 1 \leq i \leq \lfloor n/2 \rfloor + \lfloor \lceil n/2 \rceil /2 \rfloor$. Likewise, the elements that are raised to the power x_3 are such that

$$\begin{gathered}
1 \leq i \leq \lfloor \lceil \lceil n/2 \rceil /2 \rceil /2 \rfloor \\
\lfloor \lceil n/2 \rceil /2 \rfloor + 1 \leq i \leq \lfloor \lceil n/2 \rceil /2 \rfloor + \lfloor \lceil \lceil n/2 \rceil /2 \rceil /2 \rfloor \\
\lfloor n/2 \rfloor + 1 \leq i \leq \lfloor \lceil \lceil n/2 \rceil /2 \rceil /2 \rfloor \\
\lfloor n/2 \rfloor + \lfloor \lceil n/2 \rceil /2 \rfloor + 1 \leq i \leq \lfloor n/2 \rfloor + \lfloor \lceil n/2 \rceil /2 \rfloor + \lfloor \lceil \lceil n/2 \rceil /2 \rceil /2 \rfloor .
\end{gathered}$$

In general, the elements g_i that are raised to the power x_j (for $j \geq 2$) are such that $1 + k * \left\lfloor \left\lceil n2^{j-2} \right\rceil /2 \right\rfloor \leq i \leq 1 + k * \left\lfloor \left\lceil n2^{j-2} \right\rceil /2 \right\rfloor + \left\lfloor \left\lceil n2^{-j+1} \right\rceil /2 \right\rfloor$ for $0 \leq k \leq j-2$, with $k * \left\lfloor \left\lceil n2^{j-2} \right\rceil /2 \right\rfloor$ defined as $k \left\lfloor \left\lceil n2^{j-2} \right\rceil /2 \right\rfloor$ if k is odd and $k/2 * \left\lfloor \left\lceil n2^{j-3} \right\rceil /2 \right\rfloor$ if k is even and greater than 0 (if $k = 0$, it is simply 0). That is, if $v_2(k)$ denotes the dyadic valuation of k, $k * \left\lfloor \left\lceil n2^{j-2} \right\rceil /2 \right\rfloor = k2^{-v(k)} \left\lfloor \left\lceil n2^{j-2-v(k)} \right\rceil /2 \right\rfloor$.

As for $\mathbf{h}$, the elements h_i that are raised to the power x_j (for $j \geq 2$) in the expression of h are such that i is in the complement of the union of the above intervals.

Reduction to Powers of 2. This section shows to how to reduce the verification to the case in which the vector length is a power of 2. The idea is to expand, as does Algorithm Span, $\mathbf{g}$ and $\mathbf{h}$ to vectors $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ of size $2^{n'}$ by inserting $1_{\mathbb{G}}$ at specific positions so that the result of the folding procedure applied $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ are the same as the result of the procedure applied to $\mathbf{g}$ and $\mathbf{h}$ (i.e., g and h). This reduces the multi-exponentiation to the case in which the size of the vectors is a power of 2.

As a first step, the following lemma shows that the size of the vectors of group elements at the $(i+1)$th step of the protocol, which is $\lceil \lceil \lceil n/2 \rceil /2 \rceil \cdots /2 \rceil$ (i times), is actually equal to $\left\lceil n2^{-i} \right\rceil$.

```
Algorithm Span
Require: integer $n \geq 1$, vector $\mathbf{g} \in \mathbb{G}^n$
Ensure: vector $\tilde{\mathbf{g}} \in \mathbb{G}^{2^{\lceil \log n \rceil}}$
  if HammingWeight($n$) = 1 then // $n$ is a power of 2
      return $\mathbf{g}$
  end if
  $i \leftarrow$ Lsb($n$) // least significant bit of $n$ counting from 1
  $I \leftarrow \left\{ \left\lceil n2^{-i} \right\rceil + j2^{\lceil \log n \rceil - i + 1} : 0 \leq j < 2^{i-1} \right\}$
  while $i < \lfloor \log n \rfloor$ do
      $i \leftarrow i + 1$
      if $\left\lfloor \left( n \bmod 2^i \right) 2^{-i+1} \right\rfloor \bmod 2 = 0$ then // the $i$th bit of $n$ is 0
          $I \leftarrow I \cup \left\{ \left\lceil n2^{-i} \right\rceil + j2^{\lceil \log n \rceil - i + 1} : 0 \leq j < 2^{i-1} \right\}$
      end if
  end while
  $\tilde{\mathbf{g}} \leftarrow \mathbf{1}_{\mathbb{G}}^{2^{\lceil \log n \rceil}}$
  Head $\leftarrow 1$
  for $i = 1$ to $2^{\lceil \log n \rceil}$ do
      if $i \in I$ then
          continue // expand $\mathbf{g}$ by one element by inserting $1_{\mathbb{G}}$ at position $i$
      end if
      $\tilde{\mathbf{g}}[i] \leftarrow \mathbf{g}[$Head$]$
      Head $\leftarrow$ Head $+ 1$
  end for
  return $\tilde{\mathbf{g}}$
```

Lemma A.1. *For any two integers $n \geq 1$ and $i \geq 0$, $\left\lceil \left\lceil n2^{-i+1} \right\rceil / 2 \right\rceil = \left\lceil n2^{-i} \right\rceil$.*

Proof. Let w denote the Hamming weight of n. As $n \geq 1$, $w \geq 1$ as well. Then, write n as $2^{\nu_1} + \cdots + 2^{\nu_w}$, with $0 \leq \nu_1 < \cdots < \nu_w = \lfloor \log n \rfloor$, and thus $n2^{-i+1} = 2^{\nu_1 - i + 1} + \cdots + 2^{\nu_w - i + 1}$.

If $i > \lfloor \log n \rfloor = \nu_w$, then $0 < \left\lceil n2^{-i+1} \right\rceil \leq 2$, $\left\lceil \left\lceil n2^{-i+1} \right\rceil / 2 \right\rceil = 1$, and $0 < n2^{-i} = 2^{\nu_1 - i} + \cdots + 2^{\nu_w - i} < 1$, so $\left\lceil n2^{-i} \right\rceil = 1$.

If $i \leq \lfloor \log n \rfloor$, the set $\{1 \leq k \leq w \colon \nu_k > i - 1\}$ is non-empty as it always contains w. Let ℓ denotes its minimum. Rewrite then $n2^{-i+1}$ as $2^{\nu_1 - i + 1} + \cdots + 2^{\nu_\ell - i + 1} + \cdots + 2^{\nu_w - i + 1}$. Note that by definition, $\nu_\ell \geq i$.

If $\ell = 1$, then $\left\lceil n2^{-i+1} \right\rceil = 2^{\nu_\ell - i + 1} + \cdots + 2^{\nu_w - i + 1}$ and $\left\lceil \left\lceil n2^{-i+1} \right\rceil / 2 \right\rceil = 2^{\nu_\ell - i} + \cdots + 2^{\nu_w - i} = n2^{-i} = \left\lceil n2^{-i} \right\rceil$. If $\ell > 1$, further distinguish two cases:

- if $\nu_{\ell-1} = i - 1$, then $\left\lceil n2^{-i+1} \right\rceil = 2 + 2^{\nu_\ell - i + 1} + \cdots + 2^{\nu_w - i + 1}$ and $\left\lceil \left\lceil n2^{-i+1} \right\rceil / 2 \right\rceil = \left\lceil n2^{-i+1} \right\rceil / 2 = 1 + 2^{\nu_\ell - i} + \cdots + 2^{\nu_w - i}$. Besides, $n2^{-i} = 2^{\nu_1 - i} + \cdots + 2^{-1} + 2^{\nu_\ell - i} + \cdots + 2^{\nu_w - i}$, and therefore $\left\lceil n2^{-i} \right\rceil = 1 + 2^{\nu_\ell - i} + \cdots + 2^{\nu_w - i} = \left\lceil \left\lceil n2^{-i+1} \right\rceil / 2 \right\rceil$
- if $\nu_{\ell-1} > i - 1$, then $\left\lceil n2^{-i+1} \right\rceil = 1 + 2^{\nu_{\ell-1} - i + 1} + 2^{\nu_\ell - i + 1} + \cdots + 2^{\nu_w - i + 1}$ and $\left\lceil \left\lceil n2^{-i+1} \right\rceil / 2 \right\rceil = 1 + 2^{\nu_{\ell-1} - i} + \cdots + 2^{\nu_w - i} = \left\lceil n2^{-i} \right\rceil$.

□

Algorithm Fold

```
Require: integers $n \geq 2$ and $x$, vector $\mathbf{g} \in \mathbb{G}^n$
Ensure: vector $\tilde{\mathbf{g}} \in \mathbb{G}^{\lceil n/2 \rceil}$
  $\tilde{\mathbf{g}} \leftarrow \mathbf{g}_1^x \circ \mathbf{g}_2$
  return $\tilde{\mathbf{g}}$
```

Now, to prove the result of the folding procedure applied $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ are g and h, notice that it suffices to show that for any integers $n \geq 2$ and x, and any vector $\mathbf{g} \in \mathbb{G}^n$, $\textsc{Span}\left(\lceil n/2 \rceil, \textsc{Fold}(n, x, \mathbf{g})\right) = \textsc{Fold}\left(2^{\lceil \log n \rceil}, x, \textsc{Span}(n, \mathbf{g})\right)$. Indeed, in the case of the vector $\mathbf{g}$ (of size n, and recall that $n' := \lceil \log n \rceil$) at the initial protocol step and of the last base g for instance, if the above statement is true, then

$$\begin{aligned} g &= \textsc{Span}(1, g) \\ &= \textsc{Span}\left(1, \textsc{Fold}\left(2, x_{n'}, \left(\cdots \textsc{Fold}\left(n, x_1, \mathbf{g}\right) \cdots\right)\right)\right) \\ &= \textsc{Fold}\left(2, x_{n'}, \left(\cdots \textsc{Fold}\left(2^{n'}, x_1, \textsc{Span}\left(n, \mathbf{g}\right)\right) \cdots\right)\right) \\ &= \textsc{Fold}\left(2, x_{n'}, \left(\cdots \textsc{Fold}\left(2^{n'}, x_1, \tilde{\mathbf{g}}\right) \cdots\right)\right). \end{aligned}$$

To prove that $\textsc{Span}\left(\lceil n/2 \rceil, \textsc{Fold}(n, x, \mathbf{g})\right) = \textsc{Fold}\left(2^{\lceil \log n \rceil}, x, \textsc{Span}(n, \mathbf{g})\right)$ for any integers $n \geq 2$ and x, and any vector $\mathbf{g} \in \mathbb{G}^n$, distinguish the following cases:

- if n is a power of 2, then $n/2$ also is and $\textsc{Span}\left(\lceil n/2 \rceil, \textsc{Fold}(n, x, \mathbf{g})\right) = \textsc{Fold}(n, x, \mathbf{g}) = \textsc{Fold}\left(n, x, \textsc{Span}\left(n, \mathbf{g}\right)\right)$
- if n is even and not a power of 2, note that running Span on $\left(n/2, \mathbf{g}_1^x \circ \mathbf{g}_2\right)$ results in a vector of size $2^{\lceil \log(n/2) \rceil}$ with $1_{\mathbb{G}}$ at positions in

  $$\left\{\left\lceil n2^{-1}2^{-i} \right\rceil + j2^{\lceil \log(n/2) \rceil - i + 1} : 0 \leq j < 2^{i-1}\right\}$$

  for $\textsc{Lsb}(n/2) \leq i \leq \lfloor \log(n/2) \rfloor$, i.e., for integers i such that $\textsc{Lsb}(n) \leq i + 1 \leq \lfloor \log n \rfloor$.

  On the other hand,

  $$\begin{aligned} &\left\{\left\lceil n2^{-i} \right\rceil + j2^{\lceil \log n \rceil - i + 1} : 0 \leq j < 2^{i-1}\right\} \\ =& \left\{\left\lceil n2^{-1}2^{-i+1} \right\rceil + j2^{\lceil \log(n/2) \rceil - i + 2} : 0 \leq j < 2 \cdot 2^{(i-1)-1}\right\} \end{aligned}$$

  for any $\textsc{Lsb}(n) \leq i \leq \lfloor \log n \rfloor$ such that the ith bit of n is 0. Therefore, applying Fold to $\left(2^{\lceil \log n \rceil}, \textsc{Span}\left(n, \mathbf{g}\right)\right)$ results in a vector of size $2^{\lceil \log n \rceil - 1} = 2^{\lceil \log(n/2) \rceil}$ with $1_{\mathbb{G}}$ inserted in $\mathbf{g}_1^x \circ \mathbf{g}_2$ at positions

  $$\left\{\left\lceil n2^{-1}2^{-i} \right\rceil + j2^{\lceil \log(n/2) \rceil - i + 1} : 0 \leq j < 2^{(i-1)-1}\right\}$$

  for $\textsc{Lsb}(n) \leq i \leq \lfloor \log n \rfloor$. A change of variables $i \leftarrow i - 1$ then allows to conclude that $\textsc{Span}\left(n/2, \textsc{Fold}(n, x, \mathbf{g})\right) = \textsc{Fold}\left(n, x, \textsc{Span}\left(n, \mathbf{g}\right)\right)$

– if n is odd, write n as $2^0 + \cdots + 2^{\ell_1} + 2^{\nu_2} + \cdots + 2^{\nu_2+\ell_2} + \cdots + 2^{\nu_k+\ell_k}$ for $k > 1$ and $0 \le \ell_1 < \nu_2 - 1 < \cdots \le \nu_2 + \ell_2 < \cdots < \nu_k - 1 < \cdots \le \nu_k + \ell_k$. Then, $\lceil n/2 \rceil = 2^{\ell_1} + 2^{\nu_2 - 1} + \cdots + 2^{\nu_k+\ell_k-1}$. One the one hand, running algorithm Span on input $\lceil n/2 \rceil$ and $\mathbf{g}_1^x \circ \mathbf{g}_2 = \left[g_1 \cdots g_{\lceil n/2 \rceil - 1}\ 1_{\mathbb{G}}\right]^x \circ \left[g_{\lceil n/2 \rceil} \cdots g_n\right]$ returns a vector of size $2^{\lceil \log \lceil n/2 \rceil \rceil}$ with $1_{\mathbb{G}}$ inserted at positions in

$$\left\{\left\lceil \lceil n/2 \rceil 2^{-i} \right\rceil + j 2^{\lceil \log \lceil n/2 \rceil \rceil - i + 1} : 0 \le j < 2^{i-1}\right\}$$

for $i = \ell_1 + 1, \ldots, \nu_2 - 1, \ldots, \ell_{k-1} + 1, \ldots, \nu_k - 1$.

On the other hand, running algorithm Span on $(n, \mathbf{g})$ inserts $1_{\mathbb{G}}$ in $\mathbf{g}$ at positions in $\left\{\left\lceil n 2^{-i} \right\rceil + j 2^{\lceil \log n \rceil - i + 1} : 0 \le j < 2^{i-1}\right\}$ for $i = 1, \ell_1 + 2, \ldots, \nu_2, \ldots, \ell_{k-1} + 2, \ldots, \nu_k$. Consequently, running Fold on $\left(2^{\lceil \log n \rceil}, x, \textsc{Span}(n, \mathbf{g})\right)$ results in a vector of size $2^{\lceil \log n \rceil - 1} = 2^{\lceil \log \lceil n/2 \rceil \rceil}$ with $1_{\mathbb{G}}$ inserted in $\left[g_1 \cdots g_{\lceil n/2 \rceil - 1}\ 1_{\mathbb{G}}\right]^x \circ \left[g_{\lceil n/2 \rceil} \cdots g_n\right]$ at positions in

$$\left\{\left\lceil \left\lceil n 2^{-i} \right\rceil 2^{-1} \right\rceil + j 2^{\lceil \log n \rceil - (i-1)} : 0 \le j < 2^{(i-1)-1}\right\}$$

for $i = \ell_1 + 2, \ldots, \nu_2, \ldots, \ell_{k-1} + 2, \ldots, \nu_k$. Moreover, Lemma A.1 implies that $\left\lceil \left\lceil n 2^{-i} \right\rceil 2^{-1} \right\rceil = \left\lceil n 2^{-i-1} \right\rceil = \left\lceil \left\lceil n 2^{-1} \right\rceil 2^{-i} \right\rceil$, and since $\lceil \log \lceil n/2 \rceil \rceil = \lceil \log n \rceil - 1$, the change of variables $i \leftarrow i - 1$ shows that $\textsc{Span}\left(\lceil n/2 \rceil, \textsc{Fold}(n, x, \mathbf{g})\right) = \textsc{Fold}\left(2^{\lceil \log n \rceil}, x, \textsc{Span}(n, \mathbf{g})\right)$. □