Practical Statistically-Sound Proofs of Exponentiation in any Group

Charlotte Hoffmann[1], Pavel Hubáček[2], Chethan Kamath[3], Karen Klein[4], and Krzysztof Pietrzak[1]

[1]Institute of Science and Technology Austria, {charlotte.hoffmann,pietrzak}@ist.ac.at
[2]Charles University, Faculty of Mathematics and Physics, hubacek@iuuk.mff.cuni.cz
[3]Tel Aviv University, ckamath@protonmail.com
[4]ETH Zurich, karen.klein@inf.ethz.ch

Abstract

For a group $\mathbb{G}$ of unknown order, a *Proof of Exponentiation* (PoE) allows a prover to convince a verifier that a tuple $(y, x, q, T) \in \mathbb{G}^2 \times \mathbb{N}^2$ satisfies $y = x^{q^T}$. PoEs have recently found exciting applications in constructions of verifiable delay functions and succinct arguments of knowledge. The current PoEs that are practical in terms of proof-size only provide restricted soundness guarantees: Wesolowski's protocol (Journal of Cryptology 2020) is only computationally-sound (i.e., it is an argument), whereas Pietrzak's protocol (ITCS 2019) is statistically-sound only in groups that come with the promise of not having any small subgroups. On the other hand, the only statistically-sound PoE in *arbitrary* groups of unknown order is due to Block et al. (CRYPTO 2021), and it can be seen as an elaborate parallel repetition of Pietrzak's PoE. Therefore, to achieve λ bits of security, say $\lambda = 80$, the number of repetitions required – and hence the (multiplicative) overhead incurred in proof-size – is as large as λ.

In this work, we propose a statistically-sound PoE for arbitrary groups for the case where the exponent q is the product of all primes up to some bound B. For such a structured exponent, we show that it suffices to run only $\lambda/\log(B)$ parallel instances of Pietrzak's PoE. This reduces the concrete proof-size compared to Block et al. by an order of magnitude. Furthermore, we show that in the known applications where PoEs are used as a building block such structured exponents are viable. Finally, we also discuss batching of our PoE, showing that many proofs (for the same $\mathbb{G}$ and q but different x and T) can be batched by adding only a single element to the proof per additional statement.

1 Introduction

In a Proof of Exponentiation (PoE) in a group $\mathbb{G}$, a prover P aims at convincing a verifier V that a tuple $(y, x, q, T) \in \mathbb{G}^2 \times \mathbb{N}^2$ satisfies $y = x^{q^T}$, where we refer to q as the exponent and T as the time parameter. That is, it is a proof or argument system for the language

$$L_{\mathbb{G}} := \Big\{(y, x, q, T) \in \mathbb{G}^2 \times \mathbb{N}^2 : y = x^{2^T} \text{ over } \mathbb{G}\Big\}. \tag{1}$$

Note that if the order of $\mathbb{G}$, denoted $\mathrm{ord}(\mathbb{G})$, is easily computable, then V can efficiently compute x^{q^T} on its own by (i) computing the intermediate value $e = q^T$ modulo $\mathrm{ord}(\mathbb{G})$ and (ii) computing x^e using a single exponentiation in $\mathbb{G}$. Therefore, the language is non-trivial only if it is hard to compute $\mathrm{ord}(\mathbb{G})$. Such groups are referred to as *groups of unknown order*. The two concrete groups of unknown order that are presently used are RSA groups [RSA78] and class groups [BW88].[1] PoEs in such groups of unknown order have found applications as building blocks for constructing verifiable delay functions (VDFs) [Pie19, Wes20] and succinct non-interactive arguments of knowledge (SNARKs) [BHR+21]. Since both applications are non-interactive in nature, the PoE itself must be non-interactive – the usual approach is to build a *public-coin*, interactive PoE and then make it non-interactive using the Fiat-Shamir transform [FS86]. In both applications, the PoE is loosely employed in the following setting.

1. P and V get some (x, q, T) as part of the protocol and P then computes $y := x^{q^T}$ by exponentiating to exponent q sequentially[2] T times:
$$x \to x^q \to x^{q^2} \to x^{q^3} \to \ldots \to x^{q^T}. \tag{2}$$
2. P sends (y, π) to V, where π is a proof for the (true) statement $y \overset{?}{=} x^{q^T}$ computed using the non-interactive PoE.[3]

Ideally, we want the cost incurred by P to compute π in the second step to be marginal compared to the T exponentiations it makes in Equation (2) to compute y in the first place.

Importance of statistical soundness. A PoE is said to be statistically-sound if even a computationally-unbounded cheating prover $\tilde{\mathsf{P}}$ cannot convince V of a false statement $\tilde{y} \overset{?}{=} x^{q^T}$. Such a strong soundness guarantee is useful in its applications. For instance, in [BHR+21], where PoE is used to construct a SNARK, the underlying PoE must be statistically-sound so that the statistical knowledge-soundness of the SNARK can be argued. In the context of VDFs, statistical soundness ensures some security *even when* the group order is somehow revealed, e.g., in the following use-case.

[1]The RSA group, $\mathbb{Z}_N^\times$, is the (multiplicative) group of units modulo N, where $N = p_1 p_2$ is the product of two large, randomly-sampled primes p_1 and p_2. The definition of a class group is technical and requires some algebraic number theory – we refer the readers to [BW88].

[2]In VDFs, it is an explicit "sequentiality assumption" that $y = x^{q^T}$ cannot be computed faster (i.e., with fewer sequential computational steps) than as described above, even when using massive parallelism.

[3]In order to disambiguate from equality over $\mathbb{G}$, we use "$y \overset{?}{=} x^{q^T}$" throughout the paper to refer to the *statement* "$y = x^{q^T}$ holds over $\mathbb{G}$", which may or may not be true.

- Chia is a secure, permissionless blockchain [CP19], built using VDFs and proofs of space [DFKP15]. In particular, it uses a PoE-based VDF instantiated in class groups, which is sampled freshly every 10 minutes. It relies on both security properties of this VDF, i.e., (i) *sequentiality*, which (loosely-speaking) requires it to be hard to compute the output $y := x^{q^T}$ faster than in Equation (2); and (ii) *soundness*, which, as for PoE, requires it to be hard to generate proofs for a wrong output $\tilde{y} \neq x^{q^T}$. However, the reliance on these two security properties is to different degrees, as we explain next. An attacker that occasionally learns the group order and, therefore, is able to compute the output fast only has a short-term impact on the security. On the other hand, an attacker that consistently breaks soundness is devastating since it potentially leads to double-spending.[4] Therefore, if the VDF used is statistically-sound then, even in the worst case where the attacker learns the group order,[5] it will only be able to compute the correct output fast but will still *not* be able to lie about its value.

In other scenarios, where the group order is *supposed* to be known by some parties, using statistically-sound PoEs allows for a much more efficient setup. An example follows.

- The RandRunner protocol [SJH$^+$21] uses VDF to construct randomness beacons [Rab83]. Every party participating in the protocol realizing the beacon will sample two *safe primes*[6], which defines a "safe" RSA group where Pietrzak's PoE is guaranteed to have statistical soundness (see discussion below). The fact that these parties know the factorization is actually a feature, as they are occasionally required to use it as a trapdoor to *quickly* compute and broadcast a VDF output and the PoE certifying its correctness. To prevent cheating, each party must provide a zero-knowledge proof that their modulus is indeed the product of two safe primes. If one, instead, uses PoEs with statistical soundness in *arbitrary* groups, the protocol can be instantiated in *any* RSA group. Thus, the expensive zero-knowledge proof can be avoided (at the cost of larger PoEs for the individual proofs, of course).

A similar scenario comes up in the fair multiparty coin-flipping protocol of Freitag et al. [FKPS21]. This methodology might also be useful for (non-interactive) timed commitments and encryption [BN00, KLX20].

Prior works and their instantiation. The existing PoE protocols and their relevant properties – namely, soundness and proof-size – are given in Table 1. These protocols are *all* public-coin, and their non-interactive counterparts obtained by applying the Fiat-Shamir transform can be shown to be sound in the random-oracle model.[7] The PoE in

[4] A minor nuisance would be the need to roll back the blockchain once a flawed proof was added and recognized. But an attacker that can forge proofs controls the randomness and, thus, can do things like attaching a pre-computed chain to the current one in order to do a double-spending attack with only little resources.

[5] This can happen, e.g., if the trusted setup failed or the class group sampled turned out weak.

[6] A prime p is safe if $(p-1)/2$ is also prime.

[7] This follows by existing results [PS00, BCS16] for a statistically-sound protocol – in fact, the resulting non-interactive protocol is, itself, statistically-sound. However, for computationally-sound protocols, this

PoE	soundness	proof-size
[Pie19]	statistical in $\mathbb{G}$ *without* low-order elements computational in $\mathbb{G}$ with low-order elements (*low-order assumption*)	$\log(T)$
[Wes20]	computational (*adaptive root assumption*)	1
[BHR+21]	statistical in any $\mathbb{G}$	$\lambda \log(T)$

Table 1: Comparison of existing PoEs. Towards proof-size, we count the number of group elements sent by the prover – this corresponds to the size of the non-interactive proof obtained by Fiat-Shamir transform; $\lambda \in \mathbb{N}$ in the final cell denotes the statistical security parameter.

[BHR+21] is statistically-sound in *arbitrary* groups and therefore can be instantiated in both RSA groups and class groups. On the other hand, the soundness of [Pie19] and [Wes20] is computational and relies on new hardness assumptions called *low-order assumption* and, the stronger, *adaptive root assumption*, respectively [BBF18]. Therefore, these PoEs must be instantiated in groups where the respective assumption is believed to hold. An example of a group where both assumptions are believed to hold is $\mathbb{Z}_N^\times/\{\pm 1\}$, the (cyclic) group obtained by quotienting the RSA group [BBF18]. Moreover, in groups where there is a syntactic guarantee that *no* elements of low order exist, the low-order assumption would hold unconditionally and, thus, the [Pie19] PoE enjoys statistical soundness when instantiated there. An example of such a group is provided in [Pie19]: if the primes that define the RSA group $\mathbb{Z}_N^\times$ are safe primes then the subgroup of *quadratic residues* of $\mathbb{Z}_N^\times$ has no low-order elements.

For the setting of class groups, on the other hand, our understanding of the low-order and adaptive root assumptions is still only developing: for example, [BKSW20] showed how to break the low-order assumption in class groups for some classes of prime numbers and these are, therefore, not suitable to instantiate [Pie19] and [Wes20]. However, class groups do have one major advantage over RSA groups in that they do not require a trusted set-up, i.e., the set-up is "transparent': the group can be sampled *obliviously* in the sense that a random string specifies a group *without* revealing the order of the group. Compare this with the RSA group, where the only known way to generate the group is to first sample primes p_1 and p_2 and then output the modulus $N = p_1 p_2$ – but this means the sampler knows the factorization and, thus, the group order $(p_1 - 1)(p_2 - 1)$. For such "non-transparent" groups to be used in VDFs or SNARKs, one must either employ some trusted party to sample N and then delete p_1 and p_2, or sample N using expensive multiparty computation (see, e.g., [FLOP18, CCD+20] and the references therein). Thus, it is unclear how to practically instantiate the PoEs from [Pie19, Wes20] if the application requires a transparent set-up.

needs a proof: see, e.g., [BBF18].

1.1 Our Contribution

From the preceding discussion, one may conclude that the PoE of Block et al. [BHR+21] is the only reasonable option when we need statistical guarantees and want to avoid trusted set-ups. However, it suffers from the drawback that its proof-size is large, $\lambda \log(T)$ to be precise. In this work, we present an efficient PoE that enjoys statistical soundness in all groups, but only for exponent q that is of a special structured form – for our basic protocol, q is set as the product of all primes less than some *bound* $B \in \mathbb{N}$. However, the size of our proof is

$$\lambda \log(T)/\log(B),$$

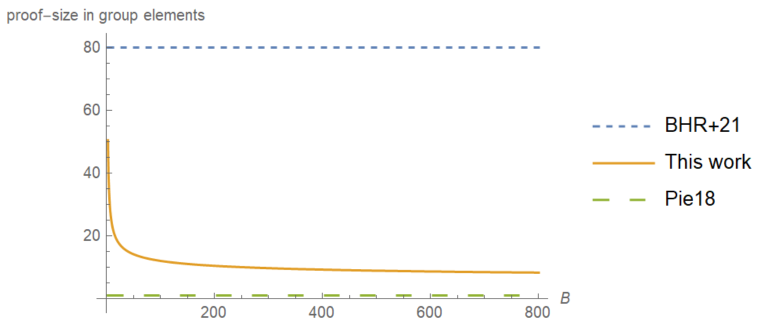
which is *smaller* than in [BHR+21] by a (multiplicative) factor of $\log(B)$. It is, however, not possible to choose B to be arbitrarily large in our protocol as this would adversely affect the verifier's (computational) complexity. An illustration of how the proof-size and verifier-complexity of our protocol change with B can be found in Figure 1. In our most basic protocol, the verifier's complexity when $B = 521$ is roughly the same as in [BHR+21] (Figure 1.(b)). For this B, we get the proof for each of the $\log(T)$ rounds down from $\lambda = 80$ to $9 = \lceil 80/\log(B) \rceil$ group elements (Figure 1.(a)). In practice, this means, e.g., that for a time parameter $T = 2^{32}$ and instantiation in a group with elements of size 2048 bits, the proof-size drops from 655KB to 74KB.

Finally, for the application to VDFs and SNARKs, we argue that our special choice of exponent q does not really matter. In the construction of SNARK in [BHR+21], it is possible to use any q as long as it is sufficiently large.[8] As for VDFs, one typically just sets the exponent $q = 2$, and exponentiation, therefore, is tantamount to squaring. For a more general q, one adjusts the time parameter accordingly, as explained next. For an arbitrary q, one can use the square-and-multiply algorithm, so each exponentiation induces $\lfloor \log(q) \rfloor$ (not just one) sequential squarings with some multiplications in-between. Note that if q was a power of 2 (which it is not in our case), say 2^k, the initial exponentiation would be of the form $x^{(2^k)^T}$, so one would set the time parameter to $T = T'/k$ in order to get a challenge that takes time T' to compute. Similarly, for our choice of q, one sets the time parameter to $T = \lceil T'/\log(q) \rceil$ to get a challenge that takes sequential time T' to compute.

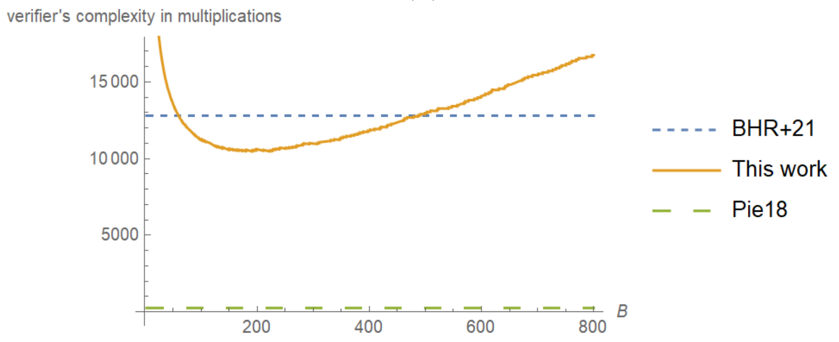
1.2 Technical Overview

In Section 1.2.1, we describe a basic version of our protocol where the verifier's running time is not optimal. However, this allows us to introduce the core ideas behind our PoE. We explain how to improve the verifier's efficiency in Section 1.2.2. We refer the readers to Sections 2 and 3, respectively, for the technical details.

[8] Note that our structured exponent q is always even. However, many results in [BHR+21] are stated only for odd values of q. In Appendix B, we show that such restriction is not necessary and that their results, in fact, hold for all values q.



(a)



(b)

Figure 1: For 80-bit security, (a) the number of (group) elements sent by the prover *per round* and (b) the number of (group) multiplications carried out by verifier, also per round, plotted for different values of the bound B. The dotted blue line, the solid orange curve, and the dashed green line represent, respectively, [BHR+21], our protocol, and [Pie19]. In Figure 3 we dissect the solid orange curve in (b).

1.2.1 Basic Protocol and Proof Idea

Our starting point is Pietrzak's PoE, in particular, an observation on the fine-grained nature of its soundness which we exploit in our protocol. Therefore, we start with its high-level description.

Pietrzak's PoE and its soundness. The protocol in [Pie19] is recursive in the time parameter T and involves $\log(T)$ rounds of interaction. To prove a (true) statement '$y = x^{q^T}$', the (honest) prover P, in the first round, sends the "midpoint" $\mu := x^{q^{T/2}}$ to the verifier V. This results in two *intermediate* statements

$$\mu \overset{?}{=} x^{q^{T/2}} \text{ and } y \overset{?}{=} \mu^{q^{T/2}}, \tag{3}$$

but relative to half the original time parameter T. Next, V sends a random challenge r to P, and they merge these two intermediate statements into a *new* statement

$$y' \overset{?}{=} (x')^{q^{T/2}}, \text{ where } x' := x^r \cdot \mu \text{ and } y' := \mu^r \cdot y.$$

The above steps constitute the "halving" sub-protocol, which is repeated $\log(T)$ times, halving the time parameter each time, until P and V arrive at a (base) statement for $T = 1$. At this point, V can efficiently check the correctness on its own by performing a single exponentiation.

To explain our observation, it suffices to focus on the first round of the protocol (but it applies also to later rounds). Assume that a cheating prover $\tilde{\mathsf{P}}$ tries to cheat with a (false) statement $\tilde{y} \overset{?}{=} x^{q^T}$ that is "α-false", by which we mean $\tilde{y} = y \cdot \alpha$ for some $\alpha \in \mathbb{G} \setminus \{\pm 1\}$ and $y := x^{q^T}$. The soundness of the halving sub-protocol depends on the order of α in $\mathbb{G}$, denoted $\operatorname{ord}(\alpha)$. For simplicity, let's restrict our attention throughout this section to the case where $\operatorname{ord}(\alpha)$ is a *prime power* p^e, for a prime $p \in \mathbb{N}$ and an exponent $e \in \mathbb{N}$ – the case of prime power already captures all interesting aspects. Our observation is that if $\operatorname{ord}(\alpha) = p^e$ for the original statement then the new statement is *still* false with probability $1 - 1/p^e$ (over the choice of r). In other words, the soundness error of this round is $1/p^e$. More generally, if $\operatorname{ord}(\alpha) = p^e$ for the original statement then for any $d \le e$, P and V derive a new statement that is α'-false with probability $1 - 1/p^d$, where $\operatorname{ord}(\alpha') = p^{e'}$ for $e' \ge e - d$. We formalise this observation in Lemma 1.

Dealing with low-order elements by parallel repetition. Note that the above analysis also explains why Pietrzak's PoE is unsound when the group contains low-order elements: when, e.g., a statement is α-false for α such that $\operatorname{ord}(\alpha) = 2$, the soundness guaranteed is only $1/2$. To generate a cheating proof, it suffices to find such an α (see [BBF18] for details of the attack) and, hence, the necessity of low-order assumption. The PoE of Block et al. [BHR+21] gets around this issue essentially by direct amplification of the weak soundness, i.e., by running λ-many instances of the PoE *in parallel*, where λ is the security parameter. We provide an overview of their protocol next.

As in Pietrzak's PoE, the protocol in [BHR+21] is recursive in the time parameter T and involves $\log(T)$ rounds of interaction. At the start of the protocol, P and V have a tuple of λ (possibly identical) statements of the form $y \overset{?}{=} x^{q^T}$, with the same exponent q and time parameter T. In the first round, for each statement $y \overset{?}{=} x^{q^T}$ in the tuple, P sends the midpoint μ to V, which results in two intermediate statements as in Equation (3). Altogether, there are 2λ intermediate statements at this point. Next, V sends a random challenge $(\mathcal{S}_1, \ldots, \mathcal{S}_\lambda)$ to P, using which the intermediate statements are randomly merged into λ *new* statements, each of the form $y \overset{?}{=} x^{q^{T/2}}$. To be precise, the challenge $\mathcal{S}_i$ determines a subset of the intermediate statements, which are multiplied to obtain the i-th new statement. The above steps constitute the halving sub-protocol for [BHR+21] repeated $\log(T)$ times, halving the time parameter T each time, until P and V arrive at a statement for $T = 1$. At this point, V can efficiently check the correctness on its own by performing λ exponentiations.

To see why elements of low order do not affect soundness of the halving sub-protocol any more, suppose that one of the original statements is α-false with, say, $\operatorname{ord}(\alpha) = 2$ (which is the worst case). Just like in [Pie19], we now have the guarantee that *at least* one of the 2λ intermediate statements is also false. [BHR+21] show that merging using the "random subset product" technique described above guarantees that each new statement is individually false with probability at least 1/2 and, by independence of choice of $\mathcal{S}_i$s, at least one of the new statements is false with probability $1 - 2^{-\lambda}$. Thus, a false statement is in some sense "propagated" till the end, at which point it gets detected by V.

Reducing the number of repetitions using structured exponents. Our basic protocol is similar to [BHR+21], except for two modifications: (i) we set q to be the product of all primes (strictly) less than some bound B; and (ii) we repeat only $\rho = \lambda/\log(B)$ times in parallel. To see why such a structured q helps reduce the number of repetitions from λ to ρ (while maintaining statistical soundness), we have to appeal to our above observation on the security of the PoE from [Pie19] and apply it to the setting of [BHR+21]. Suppose, again, that one of the original λ statements at the start of the protocol is α-false. There are three cases to be considered.

1. If $\mathrm{ord}(\alpha)$ has a "large" prime divisor $p > B$ then we apply our observation to infer that each new statement is α-false with probability $(1 - 1/p)$, and the independence of merging now implies a soundness error $p^{-\rho} \leq B^{-\rho}$. Since $\rho = \lambda/\log(B)$, we get soundness error $2^{-\lambda}$ as [BHR+21].
2. Otherwise, $\mathrm{ord}(\alpha)$ only has a "small" prime divisor $p < B$ and suppose that $\mathrm{ord}(\alpha) = p^e$.
   (a) If the exponent e of the prime power p^e is "large" – to be precise $e > C$, where $C := \log(T)\log(B)$ – then we can apply our observation again. There are $\log(T)$ rounds and, by an averaging argument, there must exist a round i such that the prime power drops by at least $\log(B)$. However, by our observation, even for $p = 2$ this can only happen with probability at most $2^{-\log(B)} = 1/B$. Now, it can be similarly argued that the probability that there exists at least one false statement in the next round is $1 - B^{-\rho}$, and we get the same soundness error as in Item 1.
   (b) Otherwise, the exponent e of the prime power p^e is "small", i.e., $e \leq C$. To handle this case, we modify the protocol so that the statement that (honest) P and V start with is
   $$y^{q^{-C}} \stackrel{?}{=} x^{q^{T-C}}$$
   instead of $y \stackrel{?}{=} x^{q^T}$, and then make V compute the final exponentiations $y^{q^{-C}} \to y$ *on its own*. To see why this helps with soundness, assume that $\tilde{\mathsf{P}}$ tries to cheat with an α-false statement $\tilde{y} \stackrel{?}{=} x^{q^{T-C}}$ in the modified protocol, i.e., $\tilde{y} = x^{q^{T-C}} \cdot \alpha$. Since $\mathrm{ord}(\alpha) = p^e$ divides q^C, we have $\alpha^{q^C} = 1$ and, therefore, V's final exponentiation leads to outright rejection:
   $$\tilde{y}^{q^C} = (x^{q^{T-C}} \cdot \alpha)^{q^C} = x^{q^T} \cdot \alpha^{q^C} = x^{q^T} \neq y.$$

To summarise our approach, the modification in Item 2b forces a cheating prover to cheat with α-false statements that are "far from being true" in the sense that $\mathrm{ord}(\alpha)$ has a divisor that is either a large prime or a prime power with large exponent. Moreover, it is possible to catch cheating with such statements building on existing techniques (Items 1 and 2a), aided by a fine-grained analysis. In Section 2, we extend the above analysis to accommodate αs of arbitrary order (Theorem 1).

1.2.2 Improving Verifier's Complexity

The basic protocol that we just outlined in Section 1.2.2 decreases the number of parallel repetitions, and, thus, the proof-size in the non-interactive case, by a factor $\log(B)$. But the verifier has to carry out some extra work as it must compute the final exponentiation $y^{q^{-C}} \to y$ on its own. This can be quite expensive, especially if we batch many proofs together. In the same group and for the same T, both protocols of Pietrzak and Block et al. can handle many PoEs basically at the price of a single PoE plus a small additive complexity overhead for each proof (this is, in fact, exploited in the SNARKs from [BHR⁺21]). In this work, we show that such batching works even for different values of T. Though, one problem for our new PoE is that, while this batching works also for the first phase of our protocol, the final exponentiation of the verifier cannot be trivially batched and, thus, it must be performed for each statement individually.

We thus further improve the protocol in two ways getting mostly rid of the extra cost for the final exponentiation. The first improvement leverages the observation that, by setting q to be not just the product of all primes (strictly) less than B but taking each prime p with power $\log(B)/\log(p)$, we can already decrease the exponent C for the final exponentiation from $\log(T)\log(B)$ to $\log(T)$. The second improvement comes from the observation that the final exponentiation $y^{q^{-C}} \to y$ can be replaced by just another PoE and, using our batching, this statement itself can be just batched together with the original statement. As the exponent ($C = \log(T)$ with the first improvement) is much smaller than T, the final exponentiation now only needs $\log(C) = \log\log(T)$ rounds. Iterating this idea $\log^*(T)$ times, which is at most

$$5 = \log^*(2^{2^{2^{2^2}}}) = \log^*(2^{65536})$$

in practice, we get the number of exponentiations down to 1 with a modest increase (from $\rho \cdot \log(T)$ to $\rho \cdot (\log(T) + \log^*(T))$ group elements) in proof-size. This batching argument only works so conveniently for T of a special form, basically powers of 2: T in the (relevant) range $2^{17} < T < 2^{65536}$ should be of the form $T = 2^t + 2^{16} + 2^4 + 2^2 + 1$. For general T the verifier's cost grows with basically the Hamming weight of $\log(T)$. In Appendix B.3 we analyse the gain in efficiency of the polynomial commitment in [BHR⁺21] when we use this improved version of our PoE as a building block instead of the PoE proposed in [BHR⁺21].

1.3 Related Work

PoE, SNARGs and VDFs. Verifiable Delay Function (VDF), as a cryptographic primitive, was first formalised in [BBBF18]. In addition to defining its security requirements, [BBBF18] provided theoretical constructions based on incrementally-verifiable computation [Val08]. Loosely speaking, they used repeated (structured) hashing as their delay function and then relied on succinct non-interactive arguments (SNARGs) to enable efficient verifiability of the result of the repeated hashing. As explained in Section 1, (non-interactive) PoE are closely related to VDFs: the practical VDFs of Pietrzak [Pie19] and Wesolowski [Wes20] use repeated squaring in a group of unknown order as their delay function and use a PoE on top to enable efficient, public verifiability of the result of the repeated squaring. The difference between [Pie19] and [Wes20] lies in the way the PoE is implemented: an overview and comparison of these PoE protocols can be found in [BBF18]. Moreover, there is evidence that to construct VDFs over groups, the reliance on the group order being unknown is inherent [RSS20, MSW20], which lends even more importance to PoE protocols from the perspective of efficient VDFs. Finally, PoE have recently been used as a crucial building block in constructing space-efficient general-purpose succinct non-interactive arguments of knowledge (SNARKs) [BFS20, BHR$^+$21, AGL$^+$22], thus establishing a converse relationship.

Additional work related to VDFs. VDFs have also been proposed in other algebraic settings: e.g., the constructions in [FMPS19, CSHT21, Sha19] are based on supersingular isogenies with the motivation to achieve (some notion of) post-quantum security.[9] In addition to the basic VDFs, refined variants of VDFs have also been explored. For a "continuous" VDF [EFKP20], it should be possible (loosely speaking) to take a proof and iterate it to produce a proof for the next iteration of the delay function (instead of having to recompute the proof for the new value from scratch). A "tight" VDF [DGMV20] necessitates that the amount of work that is required to generate a proof to be "comparable" to that required to just compute the function. Finally, we point out that existence of VDFs has implications in complexity theory, in particular to the existence of average-case hardness in complexity classes of total search problems such as **PPAD** [EFKP20, LV20, CHK$^+$19].

Timed-release cryptography. VDFs fall under the umbrella of timed-release cryptographic primitives [May94]. The first of such objects were time-lock puzzles (TLP) [RSW96] and timed commitments [BN00]. A TLP can be regarded as a delay function that also allows efficient sampling of its output (via a trapdoor). The TLP from [RSW96] uses repeated squaring in RSA group as the delay function, while the output can be efficiently determined using the factorisation of the modulus as trapdoor. Constructions of TLP are scarce – the only other known construction is from [BGJ$^+$16] and it relies on obfuscation-like assumptions. Prior to VDFs the notion of proofs of sequential work (PoSW) was introduced by Mahmoody, Moran an Vadhan [MMV13]. Like in a VDF, in a PoSW a prover on input some challenge x and time parameter T must perform an (inherently sequential) computation of $\Theta(T)$ steps and provide an efficiently verifiable proof. VDFs are a stronger notion than PoSW as in the latter the proof only certifies that a sequential computation was done, while in a VDF has an additional – for many applications crucial – "uniqueness" property, it certifies that some particular value is the correct output of a deterministic sequential computation (note that the proofs themselves need not be unique). Unlike TLPs, PoSWs can be constructed from random oracles [MMV11]. The construction from [MMV13] is based on random oracle

[9]Note that the delay functions in the RSA group and class groups of imaginary quadratic field lose their sequentiality property in the quantum setting since the order of these groups can be efficiently computed.

but is not really practical as the prover needs not just T time but also linear in T space to compute the PoSW. A construction using just $\log(T)$ space was given in [CP18], constructions with extra properties like being that "reversible" [AKK$^+$19] or "incremental" [DLM19] were recently proposed. Existing PoSW are quantum secure [BLZ21], while as mentioned above, for VDFs post-quantum security is largely open. Before practical VDFs were found, the sloth function of Lenstra and Wesolowski [LW17] was the closest we had to a unique PoSW. The reason sloth was not a unique PoSW was that verification took time linear in the time to compute the output, but verification is faster by a constant around 1000 (leveraging the difference of squaring and taking roots in groups of *known* order) and can be parallelized.

Repeated squaring. The use of repeated squaring (a special case of repeated exponentiation) in a group of unknown order as an inherently sequential operation can be traced back to [RSW96, CLSY93]. In the algebraic setting of RSA group, there is evidence that speeding up repeated squaring is tantamount to factoring [KLX20, RS20]. Further support for the sequential hardness of the problem was given in [WW20] and [vBS21]. In [FK22] Freitag and Komargodski give a lower bound for the verifier's complexity in interactive proofs for repeated squaring in the generic group model.

Batch verification. The idea of using batching to reduce the amortized cost per operation has been explored for a host of cryptographic primitives such as, e.g., key agreement [BY93], signatures [MN96], and public-key encryption [Fia97]. Closer to the topic of this paper, the problem of batching the verification of multiple *exponentiations* in arbitrary groups (not necessary of unknown order) was studied in [BGR98]. They make a heavy use of the random subset and random exponents technique (as pointed out in [Rot21]), which we also do. Building on [BGR98], Rotem [Rot21] recently explored batch-verification of VDFs: as mentioned in Section 3, Rotem focused on the verification of statements with the same time parameter, whereas our batching does not have this restriction. We refer the reader to [Rot21] for further related work on batching.

2 Basic Protocol

Block et al. [BHR$^+$21] constructed a statistically-sound PoE in any group of unknown order using the PoE from [Pie19] as starting point (which was described in Section 1.2.1). To achieve λ bits of security, their construction requires a multiplicative factor of λ in proof-size compared to [Pie19]. Below, we first explain the PoE from [BHR$^+$21] in a bit more detail (than in Section 1.2.1), and then we explain how our protocol reduces this overhead. For now we just focus on improving the proof-size, but the verifier complexity of our protocol will increase, especially in settings where we batch many proofs – later, in Section 3, we will show how to get down the verifier's complexity.

Statistical PoE from [BHR+21]. To interactively prove the statement $y \overset{?}{=} x^{q^T}$, the prover P and verifier V first make λ copies of the statement.[10] In every round of the protocol, the original statements are reduced to "smaller" statements by reducing the exponent q^{T_i} to $q^{T_{i+1}} := q^{T_i/2}$ as follows. The i-th round starts with a set of λ statements

$$\{y_i \overset{?}{=} x_i^{q^{T_i}}\}_{i\in[1,\lambda]}.$$

Then, P sends λ many "midpoints"

$$\{\mu_i := x_i^{q^{T_i/2}}\}_{i\in[1,\lambda]}$$

resulting in 2λ intermediate statements of the form

$$\{v_i \overset{?}{=} u_i^{q^{T_i/2}}\}_{i\in[1,2\lambda]}.$$

To avoid a blow-up in the number of statements, V sends a random subset $\mathcal{S} \subseteq [1, 2\lambda]$ to P and, then, P and V use $\mathcal{S}$ to recombine these 2λ intermediate statements into one using subset-product:

$$\Pi_{i\in\mathcal{S}} v_i \overset{?}{=} \Pi_{i\in\mathcal{S}} u_i^{q^{T_i/2}}.$$

To ensure soundness, it is required to perform λ many of such recombinations using independent subsets $\mathcal{S}_1, \ldots, \mathcal{S}_\lambda$, and the round ends with λ many new smaller statements. We next explain why the recombination step must be performed λ many times. Suppose only one of the 2λ intermediate statements is false before the recombination step (which is the worst case). Then, with probability $1/2$, the false statement is not among the selected statements in the random subset used during the recombination step, and the resulting new statement is true. If all new statements are true, then the verifier falsely outputs accept at the end of the protocol and, therefore, the verifier must perform λ many independent recombinations to ensure λ bit security.

Our protocol. We give a formal description of our protocol in Figure 2 – for ease of presentation, we assume that $T = 2^t + C$ for some $t \in \mathbb{N}$.[11] Moreover, since the exponent q is fixed throughout the protocol, we drop it and use the short-hand (x, y, T) to denote the statement $y \overset{?}{=} x^{q^T}$. Its soundness is then proved in Section 2.1. Below, we give a high-level overview, slightly more detailed than in Section 1.2.1. We start by listing the major differences between our protocol and [BHR+21].

1. Instead of sampling a subset $\mathcal{S} \subseteq [1, 2\lambda]$ to construct a new statement using subset-product, we take each intermediate statement to a random exponent in $\{0, 1, \ldots, 2^\kappa - 1\}$, where κ is some small integer, and then multiply them together: see Equation (5).

[10] Note that the protocol works also when the starting statements are different *as long as* the exponent q and time parameter T match. This is the case for our protocol too.

[11] The case where $T - C$ is not a power of 2 can be handled by a standard approach similar to [Pie19, Section 3.1].

2. We set
$$q := \prod_{\text{prime } p<B} p, \tag{4}$$
where B is some fixed *bound*, which can be chosen depending on the application of the PoE.

3. We define a *constant* C such that the prover gives a proof for the statement $y' \overset{?}{=} x^{q^{T-C}}$ (i.e., a q^C-th root of the original statement) and the verifier computes the final check $(y')^{q^C} = y$ itself.

The above changes allow us to reduce the number of repetitions from λ to $\rho := \lambda/\log(B)$ (for λ bits security). At a first glance, it could seem like the first change is sufficient to avoid the need for λ independent recombinations since the probability that a false statement is part of a new statement is not $1/2$ any more but seemingly $1/2^\kappa$. Unfortunately, it is not the case that taking κ-bit exponents for the recombination step achieves such a drastic improvement in the bound on the probability of accepting a false statement. Note that the process of raising a false statement to some exponent can also result in a true statement. This is indeed very likely if a false statement $y \overset{?}{=} x^{q^T}$ is "close" to the true one in the sense that y is the correct value multiplied by a low-order element α. If, e.g., this element α is of order two and the statement is raised to an even exponent, say two, the resulting statement $(y\alpha)^2 \overset{?}{=} (x^{q^T})^2$ will be true. This observation underlies an attack on [Pie19] that was first described[12] in [BBF18] and it is also the reason why [Pie19] is statistically-sound only in groups that have no elements of small order.

To circumvent the above attack using low-order elements, we introduce the second and third change in the protocol: instead of the original statement $y \overset{?}{=} x^{q^T}$, the (honest) prover only proves the (smaller) modified statement $y' \overset{?}{=} x^{q^{T-C}}$, where $y' := x^{q^{T-C}}$, and the verifier checks whether $y = (y')^{q^C}$ *by itself* as the final step. Moreover, to ensure that all the low orders are covered, we define q to be the product of all small prime numbers up to a certain bound B as in Equation (4). Now, a cheating prover that tries to cheat on an original statement by proving a false modified statement[13] *will* get caught in the final exponentiation *as long as* the false modified statement is "close" to the true one, where "close" means that the correct value can be multiplied by an element α whose order only has small prime divisors (prime numbers less than B) and the prime divisors have small exponents (integers up to C). To see this, observe that if the modified statement is $y'\alpha \overset{?}{=} x^{q^{T-C}}$ (which is false), the final exponentiation with q^C leads to rejection since
$$(\alpha y')^{q^C} = 1 \cdot (x^{q^{T-C}})^{q^C} = x^{q^T} \neq y,$$
where $\alpha^{q^C} = 1$ holds in $\mathbb{G}$ because of our assumption that it has low order. The above changes allow us to restrict to cheating provers that try to convince the verifier of statements that are

[12]The observation that random batching can be attacked using low-order elements was already made in [BP00].

[13]If the (cheating) prover does not cheat on the modified statement, the verifier will anyway catch it during the final exponentiation.

"far from true", i.e., where the correct value is multiplied by an element whose order either has a large prime divisor or a divisor which is a small prime number with a large exponent. However, in this case the probability that the protocol ends with only true statements and the verifier wrongly accepts at the end of the protocol is less than $\log(T) \cdot 2^{-\lambda}$ for parameters $C = \log(T)\log(B)$ and $\rho = \lambda/\log(B)$, where ρ takes the role of λ in [BHR+21], i.e., it is basically the number of parallel repetitions of Pietrzak's protocol.

2.1 Soundness

We show that our protocol is statistically-sound for arbitrary groups of unknown order. In particular, soundness holds against cheating provers that can compute group elements of small order.

Theorem 1. *Let B be any prime number such that $q := \prod_{\text{prime } p<B} p$ and $\rho \in \mathbb{N}$ be the number of repetitions per round. If we set $C = \log(T)\log(B)$ and let $\kappa \to \infty$, the verifier V will output* `accept` *on a false statement $(x, y, T = 2^t + C)$ with probability at most t/B^ρ.*

A parameter of our PoE is the bit-size κ of each random element sampled by the verifier. In the statement of Theorem 1, we consider the limit case with κ approaching infinity for the sake of readability. Note that if r is sampled from a randomness space of size 2^κ we have $\Pr[p \text{ divides } r] = 1/p + 1/2^\kappa$. In the limit case $\kappa \to \infty$, the probability is $1/p$. In practice, κ needs to be chosen carefully such that the protocol is still efficient but the probability of the above event is close enough to $1/p$. We discuss this point further in Section 2.2. Before proving Theorem 1, we analyse in Lemma 1 how the order of a group element precisely affects soundness; next we give provide an overview.

Fine-grained soundness. Let $x^{q^{T-C}} = y'$ but a cheating prover $\tilde{\mathsf{P}}$ claims that $x^{q^{T-C}} = y'\alpha$. In the execution of the protocol, $\tilde{\mathsf{P}}$ first sends a midpoint μ, which results in two intermediate statements. Note that no matter what the value of μ is, one of the two statements will be false, so for now let's assume that $\tilde{\mathsf{P}}$ sends a correct midpoint $\mu = x^{q^{(T-C)/2}}$. Therefore the intermediate statements are

$$\mu \overset{?}{=} x^{q^{(T-C)/2}}, \quad \text{and} \quad y'\alpha \overset{?}{=} \mu^{q^{(T-C)/2}},$$

and in particular, the second statement is α-wrong. In the protocol, we copy each statement ρ many times, raise each copy to a random exponent r_k and then multiply the 2ρ statements together. This results in a new statement that is true whenever

$$\alpha^{r_1}\alpha^{r_2}\dots\alpha^{r_\rho} = \alpha^{r_1+r_2+\dots+r_\rho} = 1.$$

This is the case when $r_1 + r_2 + \dots + r_\rho \equiv 0 \mod \operatorname{ord}(\alpha)$, which happens with probability $1/\operatorname{ord}(\alpha)$ if we assume that the randomness space is large enough (see Section 2.2 for discussion on the size of the randomness). This means that whenever $\operatorname{ord}(\alpha)$ is large, it is unlikely that the statement is transformed into a true statement after a single round.

Parameters: (determined in the analysis)

1. bound $B \in \mathbb{N}$, which defines the exponent $q := \prod_{\text{prime } p<B} p$
2. constant for exponentiation $C \in \mathbb{N}$
3. number of parallel repetitions $\rho \in \mathbb{N}$
4. size of individual random coin $\kappa \in \mathbb{N}$

Statement: $y \overset{?}{=} x^{q^T}$ in $L_{\mathbb{G}}$

Protocol: For ease of presentation, we assume that $T = 2^t + C$. The protocol consists of t rounds described in Item 2 below.

1. The prover P sends $y' = x^{q^{T-C}}$ to the verifier V, defining the initial ρ statements $\{(x_{0,j}, y_{0,j}, T_0)\}_{j\in[1,\rho]}$, where $T_0 := T - C$ and, for $j \in [1,\rho]$, $x_{0,j} := x$ and $y_{0,j} := y'$.
2. In round $i \in [1,t]$, P and V engage in the following halving sub-protocol:
   (a) Let $\{(x_{i-1,j}, y_{i-1,j}, T_{i-1} = 2^{t-i+1})\}_{j\in[1,\rho]}$ be the statement from round $i-1$.
   (b) P sends V the midpoints $\left\{\mu_{i,j} := x_{i-1,j}^{q^{T_{i-1}/2}}\right\}_{j\in[1,\rho]}$, which defines 2ρ intermediate statements
   $$\{(x_{i-1,j}, \mu_{i,j}, T_i := T_{i-1}/2)\}_{j\in[1,\rho]} \quad \text{and} \quad \{(\mu_{i,j}, y_{i-1,j}, T_i)\}_{j\in[1,\rho]},$$
   which we denote $\{(u_{i,k}, v_{i,k}, T_i)\}_{k\in[1,2\rho]}$.
   (c) V sends a random challenge $\{r_{i,j,k}\}_{j\in[1,\rho],k\in[1,2\rho]}$ to P, where $r_{i,j,k} \leftarrow \{0,1\}^\kappa$ independently for all $j \in [1,\rho]$ and $k \in [1,2\rho]$.
   (d) P and V set $\{(x_{i,j}, y_{i,j}, T_i)\}_{j\in[1,\rho]}$ as the statement for the next round, where
   $$x_{i,j} := \prod_{k\in[1,2\rho]} u_{i,k}^{r_{i,j,k}} \quad \text{and} \quad y_{i,j} := \prod_{k\in[1,2\rho]} v_{i,k}^{r_{i,j,k}}, \tag{5}$$
   and proceed to the next round.
3. V accepts if and only if $x_{t,j}^q = y_{t,j}$ and $(y')^{q^C} = y$ for all $j \in [1,\rho]$.

Figure 2: Our basic Proof of Exponentiation.

However, the order of the element that makes the statement false can also decrease round by round until the statement is transformed into a true one. To prove this intuition, we use the following well-known fact about order of group elements. A proof can be found in any standard textbook on group theory (e.g., [DF03, Proposition 5]).

Proposition 1. *Let $\mathbb{G}$ be a group, $\alpha \in \mathbb{G}$ a group element and m a positive integer. It holds that*

$$\mathrm{ord}(\alpha^m) = \frac{\mathrm{ord}(\alpha)}{\gcd(\mathrm{ord}(\alpha), m)}.$$

By Proposition 1 we get that $\mathrm{ord}(\alpha^{r_1+r_2+\cdots+r_\rho}) < \mathrm{ord}(\alpha)$ whenever $r_1 + r_2 + \cdots + r_\rho \equiv 0 \mod d$, where d is a divisor of $\mathrm{ord}(\alpha)$. If the order decreases in all of the ρ many new statements obtained this way, a cheating prover has a better chance to end up with a true statement in one of the following rounds. We want to bound the probability that after some round of the protocol all of the statements are true. To this end we need the following lemma which bounds the probability that recombining a set of $m > \rho$ statements, where at least one statement is false, gives ρ true statements. In the proof of Theorem 1 we always have $m = 2\rho$. Later in Section 3 we show how to prove many statements simultaneously so we will use the lemma with different values m.

Lemma 1. *Let $\{(x_i, y_i, T)\}_{i\in[1,m]}$ be a set of m statements such that at least one of the statements is α-false for some $\alpha \in \mathbb{G}$. Let $\{(\tilde{x}_j, \tilde{y}_j, T)\}_{j\in[1,\rho]}$ be a set of ρ statements defined as*

$$\tilde{x}_j := \prod_{i\in[1,m]} x_i^{r_{j,i}} \quad \text{and} \quad \tilde{y}_j := \prod_{i\in[1,m]} y_i^{r_{j,i}}$$

with independently sampled $r_{j,i} \leftarrow \mathbb{Z}_{2^\kappa}$ uniformly at random for all $i \in [1,m]$ and $j \in [1,\rho]$. Let B be any prime number. If we let $\kappa \to \infty$, the new statements satisfy the following properties with probability at least $1 - (1/B)^\rho$:

1. *If for some prime $p \geq B$ we have $p \mid \mathrm{ord}(\alpha)$, at least one of the statements $\{(\tilde{x}_j, \tilde{y}_j, T)\}_{j\in[1,\rho]}$ is $\tilde{\alpha}$-false and $p \mid \mathrm{ord}(\tilde{\alpha})$.*
2. *If for some prime $p < B$ and some integer $e \geq \log(B)$ we have $p^e \mid \mathrm{ord}(\alpha)$, at least one of the statements $\{(\tilde{x}_j, \tilde{y}_j, T)\}_{j\in[1,\rho]}$ is $\tilde{\alpha}$-false and $p^{e-\log(B)+1} \mid \mathrm{ord}(\tilde{\alpha})$.*

Proof. Since we want to lower bound the probabilities of the above events, it is sufficient to consider the case where $\mathrm{ord}(\alpha)$ has a single prime divisor. So, we assume $\mathrm{ord}(\alpha) = p^e$ for some prime p and integer e. Using α, we can express the statements $\{(x_i, y_i, T)\}_{i\in[1,m]}$ equivalently in the form $\{(x_i, h_i\alpha^{a_i}, T)\}_{i\in[1,m]}$, where $x_i^{q^T} = h_i$ are the correct values for all $i \in [1,m]$, $a_i \in \mathbb{Z}$ and at least one of the $a_i = 1$. A new statement $(\tilde{x}_j, \tilde{y}_j, T)$ is computed as

$$\tilde{x}_j := \prod_{i\in[1,m]} x_i^{r_{j,i}} \quad \text{and} \quad \tilde{y}_j := \prod_{i\in[1,m]} (h_i\alpha^{a_i})^{r_{j,i}}.$$

Let $\tilde{\alpha} := \prod_{i\in[1,m]} \alpha^{a_i \cdot r_{j,i}}$. By Proposition 1, the order of $\tilde{\alpha}$ is

$$\frac{p^e}{\gcd(p^e, \sum_{i=1}^m a_i r_{j,i})} = p^{e-s}$$

for some $s \in \{0, 1, \dots, e\}$. The probability that $s \geq k$ for any $k \in \{0, 1, \dots, e\}$ is

$$\Pr[s \geq k] = \Pr\left[\sum_{i=1}^m a_i r_{j,i} \equiv 0 \mod p^k\right] = \frac{1}{p^k}.$$

To prove the first claim of the lemma, we set $e = 1$ and $p = B$: the probability that the new statement is true is the probability that $s = 1$, which is $1/B$ and, therefore, the probability that all of the ρ new statements are true is $1/B^\rho$.

We prove the second claim of the lemma by setting $e \geq \log(B)$ and observing that the probability of $s \geq \log(B)$ is $1/p^{\log(B)} \leq 1/2^{\log(B)} = 1/B$ – the probability that this is the case for all ρ statements is at most $1/B^\rho$. □

Proof of Theorem 1. Assume that the correct value in Step 2 of the protocol is $y' := x^{q^{T-C}}$ but a cheating prover $\tilde{\mathsf{P}}$ claims that $y'\alpha \overset{?}{=} x^{q^{T-C}}$ (i.e., makes a statement that is α-false). Notice that in the case where $\operatorname{ord}(\alpha) \mid q^C$ we have that $(y'\alpha)^{q^C} = (y')^{q^C} = y$ and, hence, the verifier V ends up rejecting after Step 3 of the protocol. It follows that if $\tilde{\mathsf{P}}$ wants to convince V that the result is not y, then it needs to choose an element α such that $\operatorname{ord}(\alpha)$ does not divide q^C. In this case, $\tilde{\mathsf{P}}$ wins if all of the ρ statements are true after t rounds of the protocol. From the discussion above, we know that the best option for $\tilde{\mathsf{P}}$ is either picking an element of order 2^{C+1} or an element of order p, where p is the smallest prime not dividing q^C. We analyse the two cases separately.

Case 1: Let $\operatorname{ord}(\alpha) = p$. Assume that in round i of the protocol we have ρ many statements

$$\{(x_{i-1,j}, y_{i-1,j}\alpha^{a_{i-1,j}}, T_{i-1})\}_{j\in[1,\rho]} \tag{6}$$

where $a_{i-1,j} \in \mathbb{Z}$ for all $j \in [1, \rho]$. If $a_{i-1,j} \equiv 0 \mod p$, the statement is true. Otherwise it is false and, by Proposition 1 and the primality of p, we know that $\alpha^{a_{i-1,j}}$ has order p. We assume that at least one of the $a_{i-1,j}$ is not divisible by p and we bound the probability that all of the statements are true in round $i + 1$.

In Step 2 of the protocol, $\tilde{\mathsf{P}}$ sends midpoints $\mu_{i,j}$ which results in 2ρ statements

$$\{(x_{i-1,j}, \mu_{i,j}, T_i = T_i/2)\}_{j\in[1,\rho]} \text{ and } \{(\mu_{i,j}, y_{i-1,j}\alpha^{a_{i-1,j}}, T_i)\}_{j\in[1,\rho]}, \tag{7}$$

which we denote by

$$\{(u_{i,k}, v_{i,k}\alpha^{b_{i,k}}, T_i)\}_{k\in[1,2\rho]}. \tag{8}$$

Note that at least one of the $b_{i,k}$ is non-zero modulo p, no matter which elements $\mu_{i,j}$ the prover sends. Hence, the assumption of Lemma 1 is satisfied, so the probability that all of the statements in round $i+1$ are true is at most $1/B^\rho$. By the union bound, we get that the probability that all statements are true after t rounds is t/B^ρ.

Case 2: Let $\mathrm{ord}(\alpha) = 2^{C+1}$ where $C = t\ell$ for some $\ell \geq \log(B)$. In order to end up with a true statement after t rounds, $\tilde{\mathsf{P}}$ has to decrease the order of the false element by a factor of 2^{ℓ} on average per round. In particular (by an averaging argument) there has to be one round where the order decreases by at least 2^{ℓ}.

Assume that in round i of the protocol we have ρ statements given in Equation (6). Without loss of generality, let $\alpha^{a_{i-1,1}}$ have the largest order of all $\alpha^{a_{i-1,j}}$. The prover sends midpoints $\mu_{i,j}$ which results in 2ρ statements given in Equation (7) which we then denote as in Equation (8). We note that no matter the value of midpoint $\tilde{\mathsf{P}}$ sends, the order of the element that makes one of the two statements

$$\mu_{i,1} \stackrel{?}{=} x_{i-1,1}^{q^{T_i}} \quad \text{and} \quad y_{i-1,1}\alpha^{a_{i-1,1}} \stackrel{?}{=} \mu_{i,1}^{q^{T_i}}$$

false is at least $\mathrm{ord}(\alpha^{a_{i-1,1}})$. To see this, assume that $\mu_{i,1}$ is the correct midpoint but $\tilde{\mathsf{P}}$ sends $\mu_{i,1}\beta$ for some group element β. Then the second statement becomes

$$y_{i-1,1}\alpha^{a_{i-1,1}}\beta^{-q^{T_i}} \stackrel{?}{=} \mu_{i,1}^{q^{T_i}},$$

which is γ-false for $\gamma := \alpha^{a_{i-1,1}}\beta^{-q^{T_i}}$. Since $\alpha^{a_{i-1,1}} = \gamma\beta^{q^{T_i}}$ we have that $\mathrm{ord}(\alpha^{a_{i-1,1}})$ divides $\mathrm{lcm}(\mathrm{ord}(\gamma), \mathrm{ord}(\beta^{q^{T_i}}))$. It follows that $\mathrm{ord}(\alpha^{a_{i-1,1}})$ divides either $\mathrm{ord}(\gamma)$ or $\mathrm{ord}(\beta^{q^{T_i}})$ (and hence $\mathrm{ord}(\beta)$) because the order of $\alpha^{a_{i-1,1}}$ is a power of 2.

By Lemma 1, we get that the probability that none of the statements in round $i+1$ is $\tilde{\alpha}$-false, where $\tilde{\alpha}$ is some element with order divisible by $\mathrm{ord}(\alpha^{a_{i-1,1}})/2^{\ell-1}$, is at most $1/B^{\rho}$. By the union bound, we conclude that $\tilde{\mathsf{P}}$ wins after t rounds with probability at most t/B^{ρ}.

Cases 1 and 2 together yield Theorem 1. □

Corollary 1. *For $C := t\log(B)$ the Fiat-Shamir transform of our PoE yields a sound non-interactive protocol in the random-oracle model.*

Proof. As we have seen above, a cheating prover is able to convince the verifier of a false statement only if there is one round where at least one of the following two events happens depending on which attack is chosen:

- an α-false statement where $\mathrm{ord}(\alpha)$ has a prime divisor of size at least B is transformed into a true one or
- the order of the false element decreases by at least $2^{C/t}$.

We know that the probability that the output of a random oracle results in such an event is $(1/B)^{\rho}$ since, by our choice of C, we have

$$1/2^{\rho C/t} = (1/B)^{\rho}.$$

By a union bound, the probability that a cheating prover that makes up to Q queries to the random oracle will find such a query is at most $Q \cdot (1/B)^{\rho}$. □

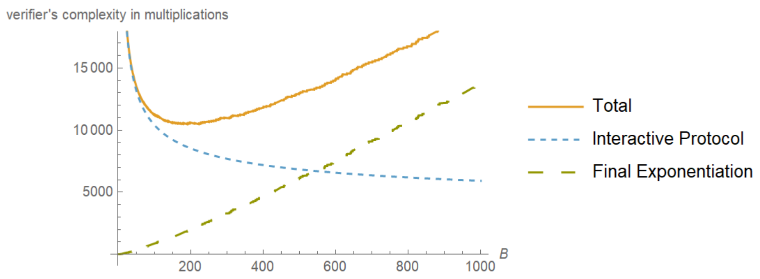


Figure 3: Number of multiplications of the verifier in one round for 80-bit security depending on the bound B. The orange solid curve is the total verifier's complexity for one round, the blue dotted graph is the cost of the interactive part of the protocol and the green dashed graph is the cost of the final exponentiation divided by the number of rounds (i.e., we amortize the cost of the final exponentiation over the number of rounds).

2.2 Efficiency

In this section, we analyse the efficiency of the Fiat-Shamir transform of our PoE for proving a statement of the form $y \overset{?}{=} x^{q^T}$ with $T = 2^t + C$.

Randomness space. In order to keep the cost of exponentiation with random coins low, we need to make the size of the randomness space as small as possible while ensuring that divisibility by B is almost uniformly distributed. For concreteness, we use $\log(B)+5$ random bits. Then it holds for any prime $p > B$ and $c \in \mathbb{Z}_p$ that

$$\Pr_{r \leftarrow \mathbb{Z}_{2^{\lceil \log(B) \rceil + 5}}}[r = c \mod p] < \frac{1}{B} + \frac{1}{B \cdot 2^5} \approx \frac{1.03}{B}.$$

Verifier's efficiency. The work for the verifier consists of two parts: 1) the interactive part, which is dominated by $t \cdot 4\rho^2$ exponentiations (with exponents of size $\log(B)+5$) and ρ exponentiations with q, and 2) the final exponentiation with q^C. Each exponentiation with a z-bit exponent via the square-and-multiply algorithm costs about $1.5z$ multiplications (i.e., z plus the Hamming weight of the exponent), so the small exponentiations have complexity $6t\rho^2(\log(B)+5)$. Additionally, the verifier performs $2t\rho^2$ multiplications to recombine the statements. The exponentiation with q^C takes $C \cdot \log(q)$ multiplications. If we set $C = t \cdot \log(B)$, the total of multiplications performed by the verifier is approximately

$$t \cdot ((6\log(B) + 32)\rho^2 + \log(B) \cdot \log(q)) + \rho \log(q) \approx t \log(B)(6\rho^2 + 2B) + 2\rho B,$$

where we use the upper bound $q \leq 4^B$ of Erdős [Erd32]. As an example, consider an implementation where $t = 32$, $B = 521$, and $\rho = \lceil 80/\log(521) \rceil = 9$. Then we have $\log(q) \approx 703$, so the cost for the verifier is around 426000 multiplications.

In Figure 3, we plot the complexity of the verifier *in a single round* of the interactive protocol for different values of B. Additionally, we consider the curves for the verifier's complexity of only the interaction with the prover and only the final exponentiation separately. Observe that, for $B < 227$, the total complexity decreases as B increases due to the fact that the number of repetitions $\lambda/\log(B)$ decreases faster than the increasing cost of the final exponentiation with q^C (the latter increases linearly with B). Beyond $B = 227$, it is the other way round and, thus, the total cost increases. Note that $B = 227$ implies $q \approx 2^{287}$. If an application requires either a value q that is much larger than this or PoEs for multiple statements (e.g., in [BHR+21], where λ many PoEs are needed in each round), then the final exponentiation of the verifier becomes too expensive. We present two modifications of the protocol that improve this complexity significantly: In Appendix A, we show how to replace $C = \log(T)\log(B)$ with $C = \log(T)$ by slightly modifying how we set q. In Section 3, we show how to compute the last step interactively without increasing the number of rounds.

Prover's efficiency. The prover needs to compute x^{q^T} and the midpoints $\mu_{i,j}$. Computing x^{q^T} takes $\log(q)\cdot T$ multiplications. If the prover stores the value $x^{q^{T/2}}$ during that computation, then computing the midpoints takes another $\rho \cdot \log(q) \cdot (T/4+T/8+\ldots+1) \approx \rho \cdot \log(q) \cdot T/2$ multiplications. This number can be significantly reduced by storing a few more elements during the computation of x^{q^T} similarly to [Pie19, Section 6.2]. For sufficiently large values of T, the cost for computing the proof can be made small compared to the cost of the T exponentiations required to compute the output and, moreover, the computation of the proof can be easily be parallelized. For this reason we mostly ignore the prover's complexity in the comparisons.

Communication complexity. The communication complexity from the prover to the verifier is of interest as it equals the proof-size after using the Fiat-Shamir heuristic. In each of the t rounds, our prover sends ρ many midpoints which are of size $\log N$. If $\log N = 2048$, $t = 32$, and $\rho = 9$ then the communication complexity is approximately 2^{19} bits.

Comparison with alternative PoEs. In Table 2, we compare our protocol with the proofs of exponentiation from [Pie19], [BHR+21], and [Wes20]. We list the proof-size and verifier's complexity. Prover's complexity is omitted since the main computation for the prover in all the protocols is dominated by the same factor, i.e., the cost of T sequential exponentiations to compute the output.

We observe that [Wes20] is the most efficient PoE regarding verifier's complexity and proof-size. However, it is not statistically-sound. [Pie19] introduces only a minor increase in overhead, but it has the drawback that it is only statistically-sound in groups with no low-order elements other than the identity. The PoE from [BHR+21] and our PoE are both statistically-sound in all groups, while the proof-size of our PoE improves by a factor of $\log(B)$ upon [BHR+21] and we compare the communication complexity per round for different values of B in Figure 1.(a).

The verifier's efficiency of our PoE depends on the choice of the bound B which also determines the size of q. In Figure 1.(b), we compare the number of multiplications per round for the verifier in both protocols for different choices of B. Additionally to the work

PoE	statistically-sound	Verifier's complexity	$\lvert\pi\rvert$
Our PoE	yes	$(6(\frac{\lambda}{\log(B)})^2 + 2B)\log(B)\log(T) + \frac{2\lambda}{\log(B)}$	$\frac{\lambda}{\log(B)}\log(T)$
[BHR$^+$21]	yes	$2\lambda^2\log(T) + 2\lambda\log(q)$	$\lambda\log(T)$
[Pie19]	in some $\mathbb{G}$	$3\lambda\log(T)$	$\log(T)$
[Wes20]	no	$\log(T) + 3\lambda$	1

Table 2: Comparison of different PoEs. Verifier's complexity is measured in the number of multiplications and proof-size $|\pi|$ in the number of group elements. By λ, we denote the statistical security parameter. [Pie19] is statistically-sound only in groups without elements of small order.

in each round, the verifier computes λ many exponentiations with q in the last round of [BHR$^+$21] and ρ many exponentiations with q in the last round of our interactive protocol. We see that the verifier's complexity improves for $B \in (59, 499)$, which corresponds to $q \in (2^{71}, 2^{685})$.

It is important to note that this is the verifier's complexity for proving a single statement. The PoE in [BHR$^+$21] achieves the same verifier's efficiency for proving λ many different statements with the same exponent simultaneously. Our protocol incurs additional $\log(T)\log(q)$ multiplications for every new statement, since the verifier has to compute the final exponentiation individually for every statement. In Section 3, we give a batching protocol that reduces the cost of the final exponentiation to $\log(q)$, which enables us to prove arbitrarily many statements simultaneously without significantly increasing the proof-size and verifier's complexity.

3 Reducing (Verifier's) Complexity by Batching

In this section, we show how to prove arbitrary many statements simultaneously without increasing the number of rounds. This batching protocol serves two purposes:

1. Efficiently proving multiple independent statements. This is needed for example in the polynomial commitment scheme of [BHR$^+$21], where in each round λ many statements need to be proven;

2. Reducing the verifier's complexity of the final exponentiation with q^C in our basic protocol. Instead of performing the computation locally, the verifier can request an additional PoE for the statement $(y')^{q^C} = y$ and verify it simultaneously with the original PoE. While now we need to do a final exponentiation for the new statement, the exponent drops from $\log(T)$ to $\log\log(T)$.

In [Rot21] Rotem gives a batching technique for arbitrary PoEs, where the statements have the same exponent. We describe a batching technique for our PoE, where the statements can have different exponents. Furthermore, the protocol can be easily adapted to the PoEs in [Pie19] and [BHR$^+$21].

3.1 The Protocol

Assume the prover wants to prove two statements in the same group $\mathbb{G}$:

$$h_1 \overset{?}{=} g_1^{q^{2^t+C_1}} \quad \text{and} \quad h_2 \overset{?}{=} g_2^{q^{2^s+C_2}}.$$

The statements can either be independent or one of them is the statement from the final verifier exponentiation of the other. The two statements can be proven simultaneously as follows. First the prover sends the statements

$$h_1' \overset{?}{=} g_1^{q^{2^t}} \quad \text{and} \quad h_2' \overset{?}{=} g_2^{q^{2^s}}.$$

We can assume that $t = \ell + s$ for some $\ell \in \mathbb{N}$. Begin with the proof of the first statement. After executing the protocol for $\ell - 1$ rounds and the prover sending midpoints in round ℓ, we have 2ρ statements

$$\left\{ v_j \overset{?}{=} u_j^{q^{2^s}} \right\}_{j \in [2\rho]}.$$

The prover makes this $2\rho + 1$ statements by adding $h_2' \overset{?}{=} g_2^{q^{2^s}}$ to them. Next the verifier sends $\rho \cdot (2\rho + 1)$ random coins and both parties create ρ new statements similarly to the original protocol. Then they proceed with the PoE protocol. Note that this process neither reduces soundness of the proof of the first statement nor of the second statement since by Lemma 1 we only need one of the statements that are being combined to be false. In the end the verifier checks if $h_1 = (h_1')^{q^{C_1}}$ and $h_2 = (h_2')^{q^{C_2}}$. This process can be extended to arbitrary-many statements of the form $h_i \overset{?}{=} g_i^{q^{2^r+C_i}}$ with the protocol given in Figure 4. Note that in Step 4 we do not specify whether the verifier checks $h_i = (h_i')^{q_i^C}$ by carrying out the computation locally or by appending it to the statements. This depends on the size of C and on the application.

Remark 1. *In the case where the exponents of q are not powers of 2, one can simply divide a statement of the form $y \overset{?}{=} x^{q^S}$ for $S \in \mathbb{N}$ into smaller statements as follows. Let $(s_0, s_1, \ldots, s_m)$ be the binary representation of S. Then we have*

$$x^{q^S} = x^{q^{\sum s_k \cdot 2^k}} = x^{\prod q^{s_k \cdot 2^k}} = y.$$

This gives at most $m + 1$ smaller statements $y_1 \overset{?}{=} x^{q^{s_0}}$ and $y_{i+1} \overset{?}{=} y_i^{q^{s_i \cdot 2^i}}$ for $i \in [1, m]$ where $y_{m+1} = y$. Again these statements can be proven simultaneously with the batching protocol.

The theorem below follows immediately from the description of the batching protocol and Remark 1.

Theorem 2. *For any $m \in \mathbb{N}$ the statements $\{(g_i, h_i, S_i + C_i)\}_{i \in [1,m]}$ can be proven in at most $1 + \max_i \log(S_i)$ rounds where additionally to one execution of the PoE protocol the following computations need to be performed:*

Parameters: Same as in Figure 2

Statements: $\left\{h_i \overset{?}{=} g_i^{q^{2^{t_i}+C_i}}\right\}_{i\in[1,m]}$ in $L_{\mathbb{G}}$ with and $t_1 > t_2 > \ldots > t_m \in \mathbb{N}$

Protocol:

1. The prover sends $h_i' := g_i^{q^{2^{t_i}}}$ for all $i \in [1, m]$ to the verifier.
2. Execute Step 2 of the PoE protocol for $(g_1, h_1', 2^{t_1})$ for $t_1 - t_2 - 1$ rounds.
3. In round $i \in [1, m-1]$ of the batching protocol we have ρ statements of the form $\{(x_j, y_j, 2^{t_{i+1}+1})\}_{j\in[1,\rho]}$:
   (a) The prover sends ρ midpoints $\{\mu_j\}_{j\in[1,\rho]}$, which results in 2ρ statements $\{(u_k, v_k, 2^{t_{i+1}})\}_{k\in[1,2\rho]}$
   (b) The prover and verifier append $(g_{j+1}, h_{j+1}', 2^{t_{i+1}})$ to the statements resulting in $2\rho + 1$ statements of the form $\{(\tilde{u}_k, \tilde{v}_k, 2^{t_{i+1}})\}_{k\in[1,2\rho+1]}$.
   (c) The verifier sends the random challenge $\{r_{j,k}\}_{j\in[1,\rho],k\in[1,2\rho+1]}$, where $r_{j,k} \in \{0,1\}^\kappa$.
   (d) They both set $\{(\tilde{x}_j, \tilde{y}_j, 2^{t_{i+1}})\}_{j\in[1,\rho]}$ as the statement for the next execution of the PoE protocol, where
   $$\tilde{x}_j := \prod_{k\in[1,2\rho+1]} \tilde{u}_k^{r_{j,k}} \quad \text{and} \quad \tilde{y}_j := \prod_{k\in[1,2\rho+1]} \tilde{v}_k^{r_{j,k}}$$
   (e) If $i < m-1$: Execute Step 2 of the PoE protocol for $t_{i+1} - t_{i+2} - 1$ rounds. Else: Execute Step 2 of the PoE protocol for t_m rounds until the statements are of the form $\{(x_j^*, y_j^*, 1)\}_{j\in[1,\rho]}$.
4. At the end of $m-1$ rounds, the verifier accepts if and only if $(x_j^*)^q = y_j^*$ for all $j \in [1, \rho]$ and $(h_i')^{q^{C_i}} = h_i$ for all $i \in [1, m]$.

Figure 4: Batching protocol for PoE.

1. *P and V perform*
$$2\rho \sum_{i=1}^{m} h(S_i)$$
additional exponentiations with exponents of size $\log(B)+5$. Here $h(S_i)$ denotes the hamming weight of S_i;

2. *V performs $m-1$ additional exponentiations with exponents q^{C_i} for $i \in [1,m] \setminus \{\arg\max_i S_i\}$;*

and the communication complexity increases by $m-1$ group elements.

Soundness of the protocol follows immediately from Lemma 1 and Theorem 1 since in the statement of Lemma 1 we consider a set of arbitrary many statements of the form (x_i, y_i, T) in any round. This means that the proof of Theorem 1 also holds when new statements are added during the execution of the protocol.

Theorem 3. *Let B be any prime number such that $q := \prod_{prime\ p<B} p$ and $\rho \in \mathbb{N}$ be the number of repetitions per round. If we set $C = \log(T)\log(B)$ and let $\kappa \to \infty$, the verifier V will output `accept` on statement $\{(g_i, h_i, 2^{t_i} + C_i)\}_{i\in[1,m]}$, where $t_1 \geq t_2 \geq \ldots \geq t_m$ and at least one statements is false, with probability at most t_1/B^ρ.*

3.2 Improving Verifier's Efficiency

In this section we analyse how the batching protocol reduces the number of multiplications for verifying a statement of the form $y \stackrel{?}{=} x^{q^T}$. In Appendix B.3 we analyse the gain in efficiency of the polynomial commitment in [BHR+21] when we use this improved version of our PoE as a building block instead of the PoE proposed in [BHR+21].

The first prover message is the value $y' = x^{q^{T-C}}$, where $C \geq \log(T)$. The key idea is that the verifier does not carry out the last exponentiation with q^C but the prover gives an interactive proof of the statement $(y')^{q^C} = y$ (a "smaller" PoE). This reduces the final exponentiation to $(y'')^{q^{C'}} = y$, where y'' is the first prover message in the smaller PoE and $C' \geq \log(C)$ is much smaller than C. This statement can again be proven interactively by an even smaller PoE. In fact, this trick can be applied recursively until the verifier only has to perform a single exponentiation with q in the final step. We make two assumptions in this section:

1. We have $q = \prod_{\text{prime } p<B} p^{\lceil \log(B)/\log(p) \rceil}$ such that the constant C in the PoE protocol is lower bounded only by $\log(T)$ and not $\log(T)\log(B)$. This is the trick we discuss in Appendix A. This assumption is needed to reduce the exponent from q^C to q and should be adopted in practice if one wants to make use of the recursion.

2. Instead of setting C to exactly $\log(T)$, we set $C = 2^{2^{2^2}} + 2^{2^2} + 2^2 + 1$, which will always be larger than $\log(T)$ in practice. This assumption is mainly for the ease of presentation and need not be adopted in practice.

Reducing the exponent from q^C to $q^{\log(C)}$. We know that exponentiation with q^C takes $C\log(q)$ multiplications. In order to reduce this cost for the verifier, we slightly modify the protocol in the following way: Instead of the verifier performing the last exponentiation locally, the verifier and the prover run the batching protocol with statements

$$\{(x, y, T = T_0 + C), (y', y, C = S_0 + C')\},$$

where $C' = \log(C)$. This modification introduces $3\rho{\cdot}h(S_0)(\log(B){+}5)$ additional multiplications during the interactive part of the protocol (by Theorem 2) *but* reduces the complexity of the final exponentiation to

$$C'\log(q) = \log(C)\log(q) \approx \log\log(T)\log(q).$$

By our special choice of C we have $h(S_0) = 1$ so we can ignore it in the remainder of the section

Applying the recursion. As we have seen, the exponent q^C can be reduced to $q^{C'}$. Now, the verifier can either perform the final exponentiation with $q^{C'}$ or apply the above procedure recursively until the verifier only has to do a single exponentiation with q in the final step. We denote the number of recursions needed until the exponent is reduced to q by $\log^*(C)$. We have that the entire recursion adds at most $3\log^*(C)\rho \cdot (\log(B) + 5)$ multiplications during the interactive part of the protocol but reduces the work of the final exponentiation from $\log(T)\log(q)$ to $\log(q)$.

In Section 2.2 we saw that the verifier's complexity without any batching is

$$\log(T) \cdot ((6\log(B) + 32)\rho^2 + \log(q)) + \rho\log(q).$$

Our batching protocol reduces the number of multiplications for verifying the proof of a single statement to approximately

$$\log(T)(6\log(B) + 32)\rho^2 + 3\log^*(C)\rho \cdot (\log(B) + 5) + (\rho + 1)\log(q)$$

and increases the proof-size to $\log^*(C) + \rho\log(T)$ group elements.

Proving multiple statements. With this optimization of the cost of verifying a single statement we can now compute the complexity of verifying m statements with our improved protocol. Each additional statement that either has exponent q^T or a smaller power of q adds $\log(q)$ multiplications to compute the final exponentiation, $3\log^*(C)\rho \cdot (\log(B) + 5)$ multiplications during the interactive part and increases the proof-size by at most $\log^*(C)$ elements. We conclude that m many statements can be proven with verifier's complexity

$$\log(T)(6\log(B) + 32)\rho^2 + 3m\log^*(C)\rho \cdot (\log(B) + 5) + (\rho + m)\log(q)$$

and communication complexity $m\log^*(C) + \rho\log(T)$.

Acknowledgements

We would like to thank the authors of [BHR+21] for clarifying several questions we had regarding their results. Pavel Hubáček was supported by the Grant Agency of the Czech Republic under the grant agreement no. 19-27871X and by the Charles University project UNCE/SCI/004. Chethan Kamath is supported by Azrieli International Postdoctoral Fellowship. Karen Klein was supported in part by ERC CoG grant 724307 and conducted part of this work at Institute of Science and Technology Austria.

References

[AGL+22] Arasu Arun, Chaya Ganesh, Satya Lokam, Tushar Mopuri, and Sriram Sridhar. Dew: Transparent constant-sized zkSNARKs. Cryptology ePrint Archive, Paper 2022/419, 2022. https://eprint.iacr.org/2022/419. 10

[AKK+19] Hamza Abusalah, Chethan Kamath, Karen Klein, Krzysztof Pietrzak, and Michael Walter. Reversible proofs of sequential work. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 277–291. Springer, 2019. 11

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, 2018. 9

[BBF18] D. Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. *IACR Cryptol. ePrint Arch.*, 2018:712, 2018. 4, 7, 10, 13

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *TCC (B2)*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016. 3

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 677–706. Springer, 2020. 10, 34, 36

[BGJ+16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 345–356. ACM, 2016. 10

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250. Springer, 1998. 11

[BHR+21] Alexander R. Block, Justin Holmgren, Alon Rosen, Ron D. Rothblum, and Pratik Soni. Time- and space-efficient arguments from groups of unknown order. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 123–152. Springer, 2021. 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 20, 21, 24, 26, 33, 34, 35, 36, 37, 38

[BKSW20] Karim Belabas, Thorsten Kleinjung, Antonio Sanso, and Benjamin Wesolowski. A note on the low order assumption in class group of an imaginary quadratic number fields. Cryptology ePrint Archive, Paper 2020/1310, 2020. https://eprint.iacr.org/2020/1310. 4

[BLZ21] Jeremiah Blocki, Seunghoon Lee, and Samson Zhou. On the security of proofs of sequential work in a post-quantum world. In Stefano Tessaro, editor, *2nd Conference on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference*, volume 199 of *LIPIcs*, pages 22:1–22:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 11

[BN00] Dan Boneh and Moni Naor. Timed commitments. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254. Springer, 2000. 3, 10

[BP00] Colin Boyd and Chris Pavlovski. Attacking and repairing batch verification schemes. In *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '00, page 58–71, Berlin, Heidelberg, 2000. Springer-Verlag. 13

[BW88] Johannes Buchmann and Hugh C. Williams. A key-exchange system based on imaginary quadratic fields. *J. Cryptol.*, 1(2):107–118, 1988. 2

[BY93] Michael J. Beller and Yacov Yacobi. Batch Diffie-Hellman key agreement systems and their application to portable communications. In Rainer A. Rueppel, editor, *Advances in Cryptology — EUROCRYPT' 92*, pages 208–220, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. 11

[CCD$^+$20] Megan Chen, Ran Cohen, Jack Doerner, Yashvanth Kondi, Eysa Lee, Schuyler Rosefield, and Abhi Shelat. Multiparty generation of an RSA modulus. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 64–93. Springer, 2020. 4

[CHK$^+$19] Arka Rai Choudhuri, Pavel Hubáček, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. PPAD-hardness via iterated squaring modulo a composite. Cryptology ePrint Archive, Report 2019/667, 2019. https://ia.cr/2019/667. 10

[CLSY93] J. . Cai, R. J. Lipton, R. Sedgewick, and A. C. . Yao. Towards uncheatable benchmarks. In *[1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference*, pages 2–11, May 1993. 11

[CP18] Bram Cohen and Krzysztof Pietrzak. Simple proofs of sequential work. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 451–467. Springer, 2018. 11

[CP19] Bram Cohen and Krzysztof Pietrzak. The Chia network blockchain. Technical report, 2019. https://www.chia.net/assets/ChiaGreenPaper.pdf, Accessed: 2022-07-29. 3

[CSHT21] Jorge Chavez-Saab, Francisco Rodríguez Henríquez, and Mehdi Tibouchi. Verifiable isogeny walks: Towards an isogeny-based postquantum VDF. Cryptology ePrint Archive, Report 2021/1289, 2021. https://ia.cr/2021/1289. 10

[DF03] David S. Dummit and Richard M. Foote. *Abstract Algebra*. John Wiley and Sons, 3rd edition, 2003. 16

[DFKP15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *CRYPTO (2)*, volume 9216 of *Lecture Notes in Computer Science*, pages 585–605. Springer, 2015. 3

[DGMV20] Nico Döttling, Sanjam Garg, Giulio Malavolta, and Prashant Nalini Vasudevan. Tight verifiable delay functions. In Clemente Galdi and Vladimir Kolesnikov, editors, *Security and Cryptography for Networks - 12th International Conference, SCN 2020, Amalfi, Italy, September 14-16, 2020, Proceedings*, volume 12238 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2020. 10

[DLM19] Nico Döttling, Russell W. F. Lai, and Giulio Malavolta. Incremental proofs of sequential work. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 292–323. Springer, 2019. 11

[EFKP20] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 125–154. Springer, 2020. 10

[Erd32] Paul Erdős. Beweis eines satzes von Tschebyschef (on a proof of a theorem of Chebyshev, in german). *Acta Litt. Sci. Szeged*, 5:194–198, 01 1932. 19

[Fia97] Amos Fiat. Batch RSA. *J. Cryptol.*, 10(2):75–88, 1997. 11

[FK22] Cody Freitag and Ilan Komargodski. The cost of statistical security in interactive proofs for repeated squaring. Cryptology ePrint Archive, Paper 2022/766, 2022. `https://eprint.iacr.org/2022/766`. 11

[FKPS21] Cody Freitag, Ilan Komargodski, Rafael Pass, and Naomi Sirkin. Non-malleable time-lock puzzles and applications. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part III*, volume 13044 of *Lecture Notes in Computer Science*, pages 447–479. Springer, 2021. 3

[FLOP18] Tore Kasper Frederiksen, Yehuda Lindell, Valery Osheter, and Benny Pinkas. Fast distributed RSA key generation for semi-honest and malicious adversaries. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 331–361. Springer, 2018. 4

[FMPS19] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th*

International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I, volume 11921 of *Lecture Notes in Computer Science*, pages 248–277. Springer, 2019. 10

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1986. 2

[KLX20] Jonathan Katz, Julian Loss, and Jiayu Xu. On the security of time-lock puzzles and timed commitments. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 390–413. Springer, 2020. 3, 11

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2010. 33

[LV20] Alex Lombardi and Vinod Vaikuntanathan. Fiat-Shamir for repeated squaring with applications to PPAD-hardness and VDFs. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 632–651. Springer, 2020. 10

[LW17] Arjen K. Lenstra and Benjamin Wesolowski. Trustworthy public randomness with sloth, unicorn, and trx. *Int. J. Appl. Cryptogr.*, 3(4):330–343, 2017. 11

[May94] Timothy C. May. Timed-release crypto, 1994. 10

[MMV11] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Time-lock puzzles in the random oracle model. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 39–50. Springer, 2011. 10

[MMV13] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Publicly verifiable proofs of sequential work. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 373–388. ACM, 2013. 10

[MN96] David M'Raïhi and David Naccache. Batch exponentiation: A fast DLP-based signature generation strategy. In Li Gong and Jacques Stearn, editors, *CCS*

'96, Proceedings of the 3rd ACM Conference on Computer and Communications Security, New Delhi, India, March 14-16, 1996, pages 58–61. ACM, 1996. 11

[MSW20] Mohammad Mahmoody, Caleb Smith, and David J. Wu. Can verifiable delay functions be based on random oracles? In *ICALP*, volume 168 of *LIPIcs*, pages 83:1–83:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. 10

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 60:1–60:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. 2, 4, 6, 7, 8, 10, 11, 12, 13, 20, 21, 33, 34

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptol.*, 13(3):361–396, 2000. 3

[Rab83] Michael O. Rabin. Transaction protection by beacons. *J. Comput. Syst. Sci.*, 27(2):256–267, 1983. 3

[Rot21] Lior Rotem. Simple and efficient batch verification techniques for verifiable delay functions. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part III*, volume 13044 of *Lecture Notes in Computer Science*, pages 382–414. Springer, 2021. 11, 21

[RS20] Lior Rotem and Gil Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 481–509. Springer, 2020. 11

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. 2

[RSS20] Lior Rotem, Gil Segev, and Ido Shahaf. Generic-group delay functions require hidden-order groups. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 155–180. Springer, 2020. 10

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1996. 10, 11

[Sha19] Barak Shani. A note on isogeny-based hybrid verifiable delay functions. Cryptology ePrint Archive, Report 2019/205, 2019. https://ia.cr/2019/205. 10

[SJH+21] Philipp Schindler, Aljosha Judmayer, Markus Hittmeir, Nicholas Stifter, and Edgar R. Weippl. Randrunner: Distributed randomness from trapdoor VDFs with strong uniqueness. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, 2021. 3

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008. 9

[vBS21] Aron van Baarsen and Marc Stevens. On time-lock cryptographic assumptions in abelian hidden-order groups. In *Advances in Cryptology – ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part II*, page 367–397, Berlin, Heidelberg, 2021. Springer-Verlag. 11

[Wes20] B. Wesolowski. Efficient verifiable delay functions. *J. Cryptol.*, 33:2113–2147, 2020. 2, 4, 10, 20, 21

[WW20] Benjamin Wesolowski and Ryan Williams. Lower bounds for the depth of modular squaring. Cryptology ePrint Archive, Report 2020/1461, 2020. https://ia.cr/2020/1461. 11

A Improving Verifier's efficiency

In Figure 1.(b) we see that for large values of B and q the verifier's complexity increases because the final computation $(y')^{q^C}$ becomes expensive. The cost of this computation is $C \cdot \log(q)$, where so far we have set $C = t\log(B)$. We can reduce this number to $C = t\log(B)/2$ by setting q to

$$q = 2^2 \cdot 3^2 \cdot \prod_{3<p<B} p. \tag{9}$$

It is straightforward to check that this does not affect our soundness bound, but it has a notable effect on verifier's efficiency as shown in Figure 5.

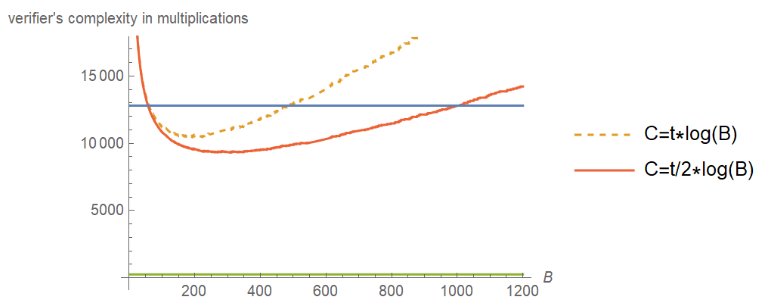


Figure 5: Number of multiplications of the verifier in one round for 80-bit security depending on the bound B. The solid blue line represents the number of multiplications in [BHR+21], the dotted orange curve represents the complexity of our protocol with $C = t\log(B)$, the solid red curve is the complexity in our protocol with $C = t\log(B)/2$ and the solid green line represents the verifier's complexity in [Pie19].

This approach can be generalized to setting $C = t\log(B)/k$ for any integer $k \leq \log(B)$. To ensure soundness we need to modify q as follows: Let m be the largest prime number such that $m < 2^k$. Then we set

$$q = 2^k \cdot 3^{\lceil k/\log(3)\rceil} \cdot 5^{\lceil k/\log(5)\rceil} \cdots m^{\lceil k/\log(m)\rceil} \cdot \prod_{m<p<B} p.$$

In particular, the choice of q that optimizes verifier's efficiency for large values of B is

$$q = \prod_{p<B} p^{\lceil \log(B)/\log(p)\rceil}$$

for which we can set $C = t$. The cost for the verifier with this parameters is shown in Figure 6. We conclude that the verifier's complexity of our scheme improves upon [BHR+21] for values of B from 59 up to 2749, which corresponds to values of q between approximately 2^{71} and $2^{400\cdot\log(2749)} \approx 2^{3167}$.

B Application in Polynomial Commitments

In this section, we discuss the application of our protocol to the polynomial commitment scheme in [BHR+21]. In particular we show in Section B.2 that one can choose the parameter q to be even and in Section B.3 we analyse the gain in efficiency when we use our PoE as a building block instead of the one proposed in [BHR+21].

A polynomial commitment scheme [KZG10] allows one party – *the committer* – to commit to a (low-degree) polynomial P. Another party – *the verifier* – can later ask the committer for an evaluation $y = P(x)$ along with a proof that helps it (efficiently) verify that the evaluation is consistent with the initial commitment c. There are two main properties that the polynomial commitment must satisfy: correctness and binding. Loosely speaking,

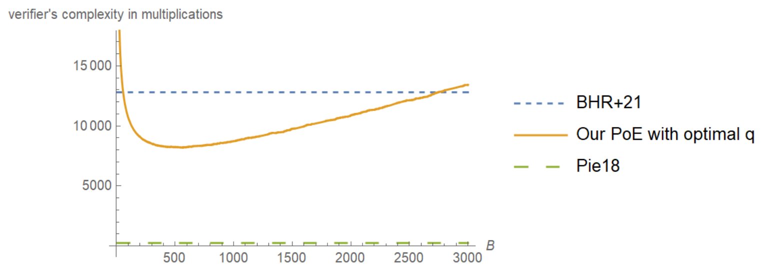


Figure 6: Number of multiplications of the verifier in one round for 80-bit security depending on the bound B. The dotted blue line represents the number of multiplications in [BHR+21], the solid orange curve is the complexity of our protocol with $C = t$ and q as above and the dashed green line is verifier's complexity in [Pie19] (which is 240 multiplications).

a polynomial commitment scheme is correct if the (honest) committer can convince the verifier of the value $y = P(x)$ of the polynomial on *any* point x on its domain, whereas it is (computationally) binding if no (computationally-bounded) cheating prover can convince the verifier of a wrong evaluation $y' \neq P(x)$. Since [BHR+21, BFS20] use their polynomial commitment scheme to build an *argument of knowledge*, they require the stronger property called *knowledge soundness* instead of just binding – i.e., the committer must know a polynomial $P(x)$ such that $y = P(x)$ and c is a commitment to $P(x)$ (formalised via an extractor). The ideas above can be naturally extended to *multilinear* polynomials.

B.1 [BHR+21] Polynomial Commitments

The time- and space-efficient (zero-knowledge) argument of knowledge from [BHR+21] is built on top of a time- and space-efficient (multilinear) polynomial commitment scheme. We first provide an overview of this polynomial commitment scheme, and then highlight the key properties that it should satisfy.

Commitment. To commit to a degree-n multilinear polynomial $P : \mathbb{F}^n \to \mathbb{F}$ over a finite field $\mathbb{F}$ of order p, the committer evaluates P over the Boolean hypercube $\{0,1\}^n$ to obtain a sequence of field elements $(z_0, \dots, z_{N-1}) \in \mathbb{F}^N$, where $N := 2^n$. This sequence is then interpreted as a sequence of digits $\mathcal{Z}$ base (large-enough) $q \in \mathbb{N}$ of an integer $z \in \mathbb{Z}$ — z is said to be the *integer encoding* of the polynomial P (see Algorithm 1). The commitment, finally, is obtained by computing the exponent $c = g^z$, where g is a random element in a group of unknown order (e.g., RSA group or class groups of imaginary quadratic field). [BHR+21] show how to carry this out using time $\tilde{O}(2^n)$ and space $\mathbf{poly}(n)$ given multi-pass streaming access to the evaluations of P on the Boolean hypercube.

Evaluation. Let $P : \mathbb{F}^n \to \mathbb{F}$ be a degree-n multilinear polynomial with integer encoding z and, for $b \in \{0,1\}$, let $P_b : \mathbb{F}^{n-1} \to \mathbb{F}$ be defined as $P(b, \cdot)$. Once committed to $c = g^z$, to prove in a verifier-efficient manner that $P(\boldsymbol{\zeta}) = \gamma$ for some $(\boldsymbol{\zeta} = (\zeta_1, \zeta_2, \ldots, \zeta_n), \gamma) \in \mathbb{F}^n \times \mathbb{F}$, the committer and the verifier proceed interactively. In the first round the committer computes, for $b \in \{0, 1\}$, $c_b := g^{z_b}$, where $z_b \in \mathbb{Z}$ is the integer encoding of P_b and $\gamma_b := P_b(\zeta_2, \ldots, \zeta_n)$. It sends $(c_0, c_1, \gamma_0, \gamma_1)$ to the verifier. The verifier checks whether

1. $\gamma = \zeta_1\gamma_1 + (1-\zeta_1)\gamma_0$ (which should hold since $P(\boldsymbol{\zeta}) = \zeta_1 P_1(\zeta_2, \ldots, \zeta_n) + (1-\zeta_1)P_0(\zeta_2, \ldots, \zeta_n)$); and

2. $c_0(c_1)^{q^{N/2}} = g^{z_0 + q^{N/2} z_1} = g^z = c$.

Note that the second equality in Item 2 relies on the homomorphic property of the integer encoding and, in turn, the commitment. Since checking $c_0(c_1)^{q^{N/2}} = c$ involves computing $(c_1)^{q^{N/2}}$ which can be expensive to the verifier, a PoE is employed to prove $c_0/c = (c_1)^{q^{N/2}}$. Now, note that checking the validity of $P(\boldsymbol{\zeta}) = \gamma$ has been reduced to checking the validity of *two* degree-$n-1$ expressions $P_b(\zeta_2, \ldots, \zeta_n) = \gamma_b$. Since recursing on both expressions is too expensive, the committer and verifier *fold* them into a single statement via random linear combination: the verifier sends a random $\alpha \in \mathbb{F}$ and the new statement is $P(\boldsymbol{\zeta}') = \gamma'$ with commitment $c' = c_0 c_1^{\alpha}$, where $\boldsymbol{\zeta}' := (\zeta_2, \ldots, \zeta_n)$ and $\gamma' = \gamma_0 + \alpha\gamma_1$. The knowledge soundness (and hence binding) of the commitment scheme is relies on the *hidden order assumption* in groups of unknown order.

Requirements from the PoE. Note that the use of the PoE in the [BHR+21] polynomial commitment is more or less black-box. However, there are two important criteria that it should satisfy.

1. Firstly, the PoE has to satisfy statistical soundness so that the knowledge soundness of the polynomial commitment built upon it can be argued ([BHR+21, Lemma 6.4]).[14] Our PoE satisfies statistical soundness.

2. Secondly, the exponent q used in the PoE protocol is borrowed *from* the polynomial commitment. In order for the polynomial commitment to satisfy its homomorphic properties, [BHR+21] set it to be a *large, odd* integer – in particular, they require $q \gg p \cdot 2^{n\mathbf{poly}(\lambda)}$. This requirement that q be large, as we saw in Section 2 is advantageous for our PoE. On the other hand, the requirement that q be odd is in conflict with our trick of choosing an *even* q as in Equation (4). However, we show in the next section that the requirement that q be odd is not necessary in [BHR+21].

B.2 Polynomial Commitments with Structured Base

Recall that the exponent q in the PoE protocol [BHR+21] is borrowed from the polynomial commitment scheme built upon it. We first observe that none of the claims pertaining to

[14] To be precise, it suffices for the soundness of the PoE to be based on a hardness assumption that is *at most* as strong as the hardness assumption that is used for showing the binding or knowledge soundness of the polynomial commitment.

the integer encoding $(\mathsf{Enc}_q, \mathsf{Dec}_q)$ in the polynomial commitment of [BHR+21] and its use in extraction rely on the exponent q being odd. In fact, the assumption in [BHR+21] that q be odd is an artefact of [BFS20] (as confirmed in a personal communication with the authors of [BHR+21]). We show in Lemmas 2 and 3 that the properties of the encoder and decoder that are necessary for the polynomial commitment of [BHR+21] to work also hold for even – and hence arbitrary – q. This allows us to use structured exponents of the form required in Section 2 (e.g., q as in Equation (4)). We first describe (for self-containment) the integer encoding from [BHR+21] in Algorithm 1 and then prove that they are consistent over $\mathbb{Z}(q/2)$ for any $q \in \mathbb{N}$ (Lemma 2). We then prove that the homomorphic properties of the decoding algorithm holds for all q (Lemma 3). Since the rest of the proofs pertaining to the polynomial commitment are unaffected by the change in exponent, [BHR+21, Theorem 4.2] can be proven also based on our PoE.

Algorithm 1 Integer encoding from [BHR+21, BFS20].

Common parameters:

1. Base $q \in \mathbb{N}$
2. Degree $n \in \mathbb{N}$ with $N := 2^n$

```
1:  procedure Enc($\mathcal{Z}$)
2:      Parse $\mathcal{Z} =: z_0, \ldots, z_{N-1} \in \mathbb{Z}(q/2)^N$
3:      return $v := \sum_{\boldsymbol{b}\in\{0,1\}^n} q^{\boldsymbol{b}} z_{\boldsymbol{b}}$
4:  end procedure

5:  procedure Dec($v$)
6:      for $k \in [0, N]$ do
7:          $S_{k-1} := v \mod q^k$
8:          if $S_{k-1} > q^k/2$ then $S_{k-1} := S_{k-1} - q^k$ end if
9:          $S_k := v \mod q^{k+1}$
10:         if $S_k > q^{k+1}/2$ then $S_k := S_k - q^{k+1}$ end if
11:     end for
12:     return $\mathcal{Z} := (z_0, \ldots, z_{N-1})$
13: end procedure
```

Lemma 2 (Bijectivity of encoder for all q, restatement of [BHR+21, Fact 5.1] and [BFS20, Fact 1])**.** *Let $q, N \in \mathbb{N}$ with $q \geq 2$. For any $v \in \mathbb{Z}(q^N/2)$, there exists a unique sequence $\boldsymbol{z} \in \mathbb{Z}(q/2)^N$ such that $v = \mathsf{Enc}_q(\boldsymbol{z})$. Furthermore, $\boldsymbol{z} = \mathsf{Dec}_q(v)$.*

Proof. The proof follows by inspection of that from [BHR+21]. That is, we argue that:

1. the domain and range have the same size; and

2. the composition of decoding and encoding functions is identity.

Since, $\mathbb{Z}(B) := \{x \in \mathbb{Z} : -B \leq x < B\}$, it follows that $\left|\mathbb{Z}(q^N/2)\right| = \left|\mathbb{Z}(q/2)^N\right| = q^N$. To show Item 2, we proceed by induction on the elements of a sequence $\mathcal{Z} := z_0, \ldots, z_{N-1} \in \mathbb{Z}(q/2)^N$.

Base case. To see that the decoder correctly recovers the first element z_0 from $\mathsf{Enc}(\mathcal{Z})$, we note that its first iteration (i.e., $k = 0$) is simply the modulo operation base q followed by a conditional shift by $q/2$. By taking the encoding function $\mathsf{Enc}(\mathcal{Z})$ modulo q, note that the higher powers of q disappear and only $z_0 \mod q$ remains. The correct representative from $[q/2, q/2)$ is then recovered by the conditional shift.

Induction hypothesis. Assume that the first k elements $z_0, \ldots, z_{k-1}$ have been correctly recovered. In particular, this implies the sequence $S_{-1}, S_0, \ldots, S_k$ has been correctly computed.

Induction. To see that the decoder correctly recovers z_{k+1}, we take $\mathsf{Enc}(\mathcal{Z})$ modulo q^{k+1}. Since S_k has been correctly computed, and since $z_{k+1}q^k + S_k = v = S_{k+1} \mod q^{k+1}$, it follows from the description of the decoder (i.e., $z_{k+1} := (S_{k+1} - S_k)/q^k$) that it recovers the correct representative of z_{k+1} after the conditional shift.

□

Lemma 3 (Homomorphism of decoder for all q, restatement of [BHR+21, Claim 5.2])**.** *Let $q, N \in \mathbb{N}$ with $q \geq 2$. Also let $\ell \in \mathbb{N}$ and $B_1, B_2 \geq 1$ be such that $B_1 \cdot B_2 \leq q/(2\ell)$. Then, for every $a_1, \ldots, a_\ell \in \mathbb{Z}(B_1)$, and integers $z_1, \ldots, z_\ell \in \mathbb{Z}(q^N/2)$ such that $\mathsf{Dec}_q(z_i) \in \mathbb{Z}(B_2)^N$,*

$$\mathsf{Dec}_q\left(\sum_{i\in[1,\ell]} a_i \cdot z_i\right) = \sum_{i\in[1,\ell]} a_i \cdot \mathsf{Dec}_q(z_i) \tag{10}$$

Sketch. Once Lemma 2 has been reproved for even q, the argument is the same as in[BHR+21]. That is, one argues that:

1. Encoding of LHS and RHS in Equation (10) are equal; and
2. Encoding of LHS and RHS in Equation (10) are in $\mathbb{Z}_q^N$. □

B.3 Efficiency

In this section we analyse the improvement in efficiency of the polynomial commitment scheme in [BHR+21] using our PoE, the batching protocol and the optimization in Appendix A. In the polynomial commitment scheme the PoE protocol is used to prove statements of the form $x_i^{q^{2^{n-k-1}}} = y_i$ for every $i \in [\lambda]$ and every $k \in \{0, 1, \ldots, n-1\}$.

Communication complexity. In [BHR+21] the communication complexity of proving λ many statements with the same exponent is $\lambda(n-k-1)$ group elements. This gives a total PoE proof-size of

$$\lambda\sum_{k=0}^{n-1}(n-k-1)=\frac{\lambda}{2}(n-1)n.$$

As we have seen in Section 3.2, in our PoE the cost of proving λn statements, in which the largest exponent is q^{n-1}, is

$$\lambda n\log^*(n-1)+\frac{\lambda}{\log(B)}(n-1).$$

We conclude that we decrease the proof-size of the polynomial commitment by a factor of approximately $n/(2\log^*(n-1))$. This number can be increased to $n/2$ at the cost of a higher verifier complexity. More generally, the number of recursive steps explained in Section 3.2 can be used to choose a trade-off between proof-size and verifier efficiency.

Verifier's efficiency. In [BHR+21] the verifier's complexity of proving λ many statements with the same exponent is $2\lambda^2(n-k-1)+\lambda\log(q)$ multiplications. This gives a total verifier's complexity of

$$2\lambda^2\sum_{k=0}^{n-1}((n-k-1)+\lambda\log(q))=(\lambda\log(q)+2\lambda^2(n-1))n.$$

As we have seen in Section 3.2, in our PoE the cost of verifying λn statements, in which the largest exponent is q^{n-1}, is

$$(n-1)(6\log(B)+32)\rho^2+3\lambda n\log^*(C)\rho\cdot(\log(B)+5)+(\rho+\lambda n)\log(q)\approx 15\lambda^2 n+\lambda n\log(q).$$

Since in practice we have $n\approx 32$, we conclude that the verifier's efficiency of the polynomial commitment scheme implemented with our PoE is comparable to that in [BHR+21].