Practical Statistically-Sound Proofs of Exponentiation in any Group*

Charlotte Hoffmann[0000−0003−2027−5549]1, Pavel Hubáček[0000−0002−6850−6222]2, Chethan Kamath[3], Karen Klein[4], and Krzysztof Pietrzak[1]

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

Abstract. A proof of exponentiation (PoE) in a group $\mathbb{G}$ of unknown order allows a prover to convince a verifier that a tuple $(x, q, T, y) \in \mathbb{G} \times \mathbb{N} \times \mathbb{N} \times \mathbb{G}$ satisfies $x^{q^T} = y$. This primitive has recently found exciting applications in the constructions of verifiable delay functions and succinct arguments of knowledge. The most practical PoEs only achieve soundness either under computational assumptions, i.e., they are arguments (Wesolowski, Journal of Cryptology 2020), or in groups that come with the promise of not having any small subgroups (Pietrzak, ITCS 2019). The only statistically-sound PoE in *general* groups of unknown order is due to Block et al. (CRYPTO 2021), and can be seen as an elaborate parallel repetition of Pietrzak's PoE: to achieve λ bits of security, say $\lambda = 80$, the number of repetitions required (and thus the blow-up in communication) is as large as λ.

In this work, we propose a statistically-sound PoE for the case where the exponent q is the product of all primes up to some bound B. We show that, in this case, it suffices to run only $\lambda/\log(B)$ parallel instances of Pietrzak's PoE, which 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.

* 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.

1 Introduction

In a proof of exponentiation (PoE) in a group $\mathbb{G}$, a prover $\mathcal{P}$ aims at convincing a verifier $\mathcal{V}$ that a tuple $(x, q, T, y) \in \mathbb{G} \times \mathbb{N} \times \mathbb{N} \times \mathbb{G}$ satisfies $x^{q^T} = y$. Note that such proofs are only of interest if the order $\text{ord}(\mathbb{G})$ of $\mathbb{G}$ is not known: otherwise, one can efficiently compute x^{q^T} by first computing the exponent modulo the group order, i.e., $e = q^T \bmod \text{ord}(\mathbb{G})$, and then computing x^e using a single exponentiation x^e in $\mathbb{G}$.

PoEs in groups of unknown order have found applications for constructing verifiable delay functions (VDFs) [40,50] and as building blocks for time- and space-efficient succinct non-interactive arguments of knowledge (SNARK) [7]. In these applications, the prover and verifier get (x, q, T) and then $\mathcal{P}$ computes x^{q^T} by exponentiating to the power q sequentially[5] T times:

$$x \to x^q \to x^{q^2} \to x^{q^3} \to \ldots \to x^{q^T}.$$

In the next step, $\mathcal{P}$ sends y to $\mathcal{V}$ and then they run an interactive protocol where $\mathcal{P}$ convinces $\mathcal{V}$ that $y = x^{q^T}$. The existing protocols are all public-coin and, thus, can be made non-interactive in the random-oracle model using the Fiat-Shamir heuristic [27].

Soundness of PoEs. In the PoEs mentioned above, the prover's computation for the proof is marginal compared to the T exponentiations required to compute y in the first place, but the proofs differ in size. As illustrated in Table 1, in [50] the proof is just one group element, in [40] it is $\log(T)$ elements and in [7] it is $\lambda \log(T)$ elements for a statistical security parameter λ.

On the other hand, [7] is statistically-sound (and the non-interactive proof inherits this security in the random oracle model), while the soundness of [50] relies on a new computational hardness assumption called *adaptive root assumption*. Like with the proof-size, [40] lies in-between the other two protocols also in terms of the assumptions required for its soundness. It relies on the *low order assumption*, which requires that it is hard to find a (non-identity) element with low order in $\mathbb{G}$. This assumption is weaker than the adaptive root assumption [10] and, in groups where no low order elements exist, it holds unconditionally and, thus, the [40] PoE has statistical soundness.

The two concrete groups of unknown order that have been suggested are RSA groups [41] and class groups of imaginary quadratic fields [13]. An RSA group $\mathbb{Z}_N^*$ is defined by a product $N = p \cdot q$ of two large randomly sampled primes p, q. In [40], it was observed that if p, q are chosen to be safe primes[6] then the subgroup of quadratic residues of $\mathbb{Z}_N^*$ has no low order elements and, thus, the PoE is statistically-sound.

[5] 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.

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

While class groups are much less studied than RSA groups, they have one major advantage, explained next. The only known way to sample an RSA group is to first sample p, q and then output $N = p \cdot q$, but this means the sampler knows the factorization and thus the group order $(p-1)(q-1)$. For such groups to be used for VDFs or SNARKs, one thus needs to either employ some trusted party to sample N and truthfully delete p, q, or sample N in an expensive multiparty computation (see, e.g., [28,17] and the references therein). Class groups on the other hand have a "transparent" setup: they can be sampled obliviously in the sense that a random string specifies a group without revealing the order of the group. However, our understanding of non-standard assumptions, like the low-order assumption, is still developing in class groups: in 2020 the authors of [3] showed how to break the low-order assumption in class groups for some classes of prime numbers.

Why Statistical Soundness? Recall that the only statistically-sound PoE in a group with transparent setup is from [7]. There, the PoE is used in a proof of knowledge and, to argue statistical knowledge-soundness of the protocol, the underlying PoE must be statistically-sound.

Also when a PoE is used in VDFs, statistical soundness can be crucial as such VDFs still provide some security even when the group order is revealed. Moreover, in settings where the group order is supposed to be known by some parties, it allows for a much more efficient setup. We discuss those two settings below.

Recall that a VDF has two security properties: the first is the sequentiality, which states that the output $y := x^{q^T}$ cannot be computed faster than by T sequential exponentiations; the second is the soundness of the proof certifying that y is the correct output. If a VDF is statistically-sound then, even in the worst case where an attacker learns the group order (say because the trusted setup failed, or in the case of sampling a weak class group), the attacker will only be able to compute the output fast but it will still not be able to lie about its value. In a design like Chia (`chia.net`), which combines VDFs with proofs of space to get a secure permissionless blockchain, an attacker that occasionally learns the group order (Chia uses class groups which are sampled freshly every 10 minutes) has limited impact on the security, but breaking the soundness of the VDF could be potentially devastating.[7]

Statistically-sound VDFs have also been used to construct randomness beacons like in the RandRunner protocol [46]. Their setup is not transparent: every party participating in the protocol realizing the beacon will sample two safe primes which then can be used in Pietrzak's statistically-sound PoE. The fact that these parties know the factorization is actually a feature, as they are occasionally required to use it as a trapdoor to compute and broadcast a VDF output

[7] 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.

and the PoE certifying its correctness fast. To prevent parties from lying, they must provide a zero-knowledge proof that their modulus is the product of two safes primes. Using the statistically-sound PoE from this work, we can avoid this expensive ZK proof and just use any RSA modulus, at the cost of larger PoEs for the individual proofs.

Generally, by using a VDF that is statistically-sound in any group allows us to skip the expensive zero-knowledge proof showing that a group was sampled correctly during setup (i.e., it has no low order elements) for protocols where statistical soundness is required because the party sampling the group knows the group order and, thus, could easily break soundness otherwise. Apart from randomness beacons as RandRunner, a related scenario comes up in the fair multiparty coin-flipping protocol of Freitag et al. [30]. This methodology might also be useful for (non-interactive) timed commitments [11] or encryption [32].

1.1 Our Contribution

As outlined above, Wesolowski's PoE has proofs of size one (group element) under the *adaptive root assumption*, Pietrzak's PoE has proofs of size $\log(T)$ under the weaker *low order assumption*, and the PoE of Block et al. has proofs of size $\log(T)\cdot\lambda$ for a statistical security parameter λ, say $\lambda = 80$. The protocol of Block et al. is the only PoE with statistical soundness in a group with transparent setup.

In this work, we present a new PoE to certify that (x, q, T, y) satisfies $y = x^{q^T}$ with statistical soundness in all groups. Our PoE only works for q of a special form. Namely, q is the product of all primes less than some bound B and, for such q, we get a proof-size $\log(T)\cdot\lambda/\log(B)$, i.e., by a factor $\log(B)$ smaller than in Block et al. [7]. Fortunately for the applications to VDFs or SNARKs discussed above, the choice of q does not really matter: in the SNARKs application [7] one can use any q that is sufficiently large.[8] For VDFs, one typically just sets $q = 2$, so exponentiation means one squaring. Having a more general q we can use square and multiply, so each exponentiation are $\lfloor\log(q)\rfloor$ (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.

We cannot choose B too large, as a larger B negatively affects the verifier's complexity. As illustrated in Figure 2, in our most basic protocol, the verifier's complexity is roughly the same as in Block et al. for $B = 521$. For this B, we get the proof down from $\lambda = 80$ to $9 = \lceil 80/\log(B)\rceil$ elements for each of the $\log(T)$ rounds as illustrated in Figure 1. In practice, this means that, e.g., for $T = 2^{32}$

[8] In [7] many results are stated only for odd choices of q. In Appendix B we show that they also hold for even q.

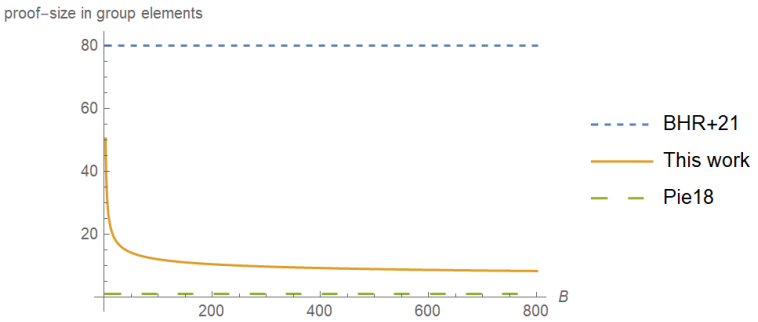

Fig. 1. Number of elements sent by the prover in one round for 80-bit security depending on the bound B. The dotted blue line is the proof-size in [7], the orange graph is the proof-size in our protocol and the dashed green line is the proof-size in [40] (which is one element per round).

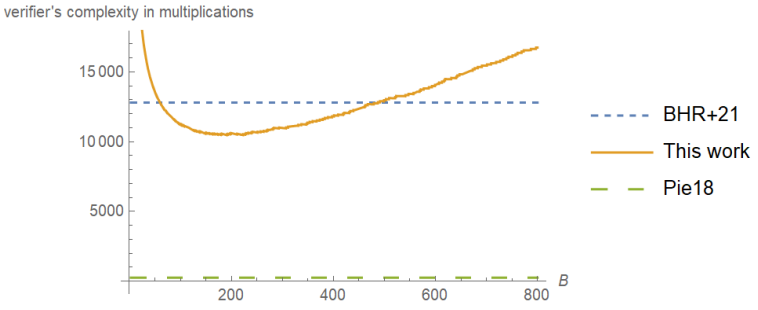

Fig. 2. Number of multiplications of the verifier in one round for 80-bit security depending on the bound B. The dotted blue line is the number of multiplications in [7], the orange graph is the number of multiplications in our protocol and the dashed green line is the verifier's complexity in [40]. In Figure 4 we dissect the orange curve.

and a group with elements of size 2048 bits, the proof-size drops from $655KB$ to $74KB$.

Basic Protocol and Proof Idea. Our starting point is the following observation on the soundness in Pietrzak's PoE: Pietrzak's protocol proceeds in $\log(T)$ rounds, where each round starts with a claim $x^{q^T} \stackrel{?}{=} y$ and ends with a claim $y' \stackrel{?}{=} x'^{q^{T/2}}$ for a T of half the size. Assume that, at the beginning of a round, we have the wrong claim that $y' \stackrel{?}{=} x^{q^T}$ while $y = x^{q^T}$, where $y' = y \cdot \alpha$ with $\alpha \notin \{1, -1\}$. The soundness of the protocol then depends on the order $\mathrm{ord}(\alpha)$ of α (i.e., $\alpha^{\mathrm{ord}(\alpha)} = 1$). Concretely, if $\mathrm{ord}(\alpha) = p^e$ for some prime power e of p at the beginning of a round (in this introduction, we only consider the special case

of a single prime power as it already contains all interesting aspects) then the claim at the end of the round is still wrong with probability $1 - 1/p^e$ (i.e., this round has a soundness error of $1/p^e$). More generally, for any $t \leq e$, we end up with a claim for some $y \cdot \alpha'$ instead of the correct y with probability $1 - 1/p^t$, where $\text{ord}(\alpha') = p^{e'}$ for $e' \geq e - t$.

Note that this means that Pietrzak's protocol is statistically-sound if no low order elements exist. The PoE by Block et al. does not need any assumption about $\text{ord}(\alpha)$, and it achieves statistical soundness even if $\text{ord}(\alpha) = 2$ (while Pietrzak's PoE is only 1/2 sound for such α) by basically running λ PoEs in parallel. In each round, one starts with λ claims of the form $x^{q^T} \stackrel{?}{=} y$ and, for each claim, the prover provides μ which it claims is the "midpoint" satisfying both $x^{q^{T/2}} \stackrel{?}{=} \mu, \mu^{q^{T/2}} \stackrel{?}{=} y$. At this point, we have 2λ claims, at least one of which is wrong if one of the original claims was wrong. These 2λ claims are then randomly combined into λ claims of the form $x^{q^{T/2}} \stackrel{?}{=} y$. Each of these claims is individually wrong with probability 1/2 and, thus, at least one of them is wrong with probability $1 - 2^{-\lambda}$. Each round gets the exponent in the claims down from T to $T/2$ and, after $\log(T)$ rounds, we have claims that the verifier can efficiently verify itself with a single exponentiation.

In our protocol, we use a similar strategy as Block et al.: We run ρ PoEs in parallel (where ρ can be smaller than the statistical parameter λ). Unlike Block et al., we require q to be of a special form, in our basic protocol it is the product of all primes less than some bound B. If $\text{ord}(\alpha)$ has a prime divisor $p > B$ then we use the same security argument as above (but with p not 2) to get soundness error $p^{-\rho} \leq B^{-\rho}$, in this case we get soundness $2^{-\lambda}$ as Block et al. with only $\rho \approx \lambda / \log(B)$ instead of λ repetitions. Otherwise, we have $\text{ord}(\alpha) = p^e$ for some $p < B$. If the prime power e is large, concretely $e \geq \log(T)\log(B)$ then we again can basically use the argument above. In each of the $\log(T)$ rounds, the prime power must go down by $\log(B)$ on average, and even for $p = 2$ that only happens with prob $2^{-\log(B)} = 1/B$.

Therefore, we are left with the case $\text{ord}(\alpha) = p^e$ with $e \leq C = \log(T)\log(B)$. To handle this case, we change the statement to be proven from $y \stackrel{?}{=} x^{q^T}$ to $y' \stackrel{?}{=} x^{q^{T-C}}$ and we let the verifier compute the final $y = y'^{q^C}$ itself. Assume that the prover wrongly claims $y'' \stackrel{?}{=} x^{q^{T-C}}$ with $y'' = y' \cdot \alpha$. With α as above, the final exponentiations of the verifier eliminate α. Since now the order p^e of α divides q^C, we have that $\alpha^{q^C} = 1$ and, so,

$$y''^{q^C} = (y' \cdot \alpha)^{q^C} = y'^{q^C} \cdot \alpha^{q^C} = y'^{q^C} = y.$$

Improving the Verifier's complexity. The basic protocol that we just outlined 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 $(y', q, C) \to y'^{q^C}$ *by itself.* 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 [7]). 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 $< B$ 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 $(y', q, C) \to y'^{q^C}$ 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.1 we analyze the gain in efficiency of the polynomial commitment in [7] when we use this improved version of our PoE as a building block instead of the PoE proposed in [7].

1.2 Additional Related Work

PoE, SNARGs and VDFs. Verifiable Delay Function (VDF), as a cryptographic primitive, was first formalised in [9]. In addition to defining its security requirements, [9] provided theoretical constructions based on incrementally-verifiable computation [48]. 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 [40] and Wesolowski [50] 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 [40] and [50] lies in the way the PoE is implemented: an overview and comparison of these PoE protocols can be found in [10]. Moreover, there is evidence that to construct VDFs over groups, the reliance on the group order being unknown is inherent [45,37], 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) [14,7,2], thus establishing a converse relationship.

Additional related work to VDFs. VDFs have also been proposed in other algebraic settings: e.g., the constructions in [25,16,47] 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 [23], 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 [20] 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** [23,34,18].

Timed-release cryptography. VDFs fall under the umbrella of timed-release cryptographic primitives [38]. The first of such objects were time-lock puzzles (TLP) [42] and timed commitments [11]. A TLP can be regarded as a delay function that also allows efficient sampling of its output (via a trapdoor). The TLP from [42] 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 [6] 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 [36]. 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. Unlike TLPs, PoSWs can be constructed from random oracles (RO) [35]. The construction from [36] is based on ROs 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 [19], constructions with extra properties like being that "reversible" [1] or "incremental" [21] were recently proposed. Existing PoSW are quantum secure [8], while as mentioned above, for VDFs post quantum security is largely open. Before practical VDFs were found, the sloth function of Lenstra and Wesolowski [33] 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.

[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.

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 [42,15]. In the algebraic setting of RSA group, there is evidence that speeding up repeated squaring is tantamount to factoring [32,44]. Further support for the sequential hardness of the problem was given in [51] and [49]. In [29] 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 [5], signatures [39], and public-key encryption [26]. Closer to our topic, the problem of batching the verification of multiple *exponentiations* in arbitrary groups (not necessary of unknown order) was studied in [4]. They make a heavy use of the random subset and random exponents technique (as pointed out in [43]), which we also do. Building on [4], Rotem [43] 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 [43] for further related work on batching.

2 Basic Protocol

Block et al. [7] constructed a statistically-sound PoE in any group of unknown order using the PoE from [40] as starting point. To achieve λ bits of security, their construction requires a multiplicative factor of λ in proof-size compared to [40]. Below, we first explain the PoE from [7] in a bit more detail (than Section 1.1), 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 [7]. To interactively prove the statement $x^{q^T} \stackrel{?}{=} y$, the prover and verifier first make λ copies of the statement. In every round of the protocol, the original claims 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 $\{x_i^{q^{T_i}} \stackrel{?}{=} y_i\}_{i\in[1,\lambda]}$. The prover then sends λ many "midpoints" $\{\mu_i := x_i^{q^{T_i/2}}\}_{i\in[1,\lambda]}$ resulting in 2λ statements of the form $\{u_i^{q^{T_i/2}} \stackrel{?}{=} v_i\}_{i\in[1,2\lambda]}$. To avoid a blow-up in the number of statements in every round, the verifier recombines these 2λ statements by taking a random subset of them and multiplying the statements in the subset together, i.e., obtaining a *single* statement. To ensure soundness, the verifier performs λ many of such recombinations independently and the round ends with λ many new smaller statements. It is easy to see why the recombination step must be performed λ many times: Suppose only one of the

2λ statements is incorrect before the recombination step. Then, with probability $1/2$, the incorrect statement is not chosen among the statements in the random subset used during the recombination step and the resulting new statement is correct. If all new statements are correct, 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. In this work, we improve the efficiency of the above PoE by introducing the following changes in the protocol:

1. Instead of sampling a subset to construct a new statement, we take each statement to a random exponent in $\{0, 1, \ldots, 2^\kappa - 1\}$, where κ is some small integer, and then multiply them together.
2. We set
$$q := \prod_{\text{prime } p<B} p, \tag{1}$$
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 $x^{q^{T-C}} \stackrel{?}{=} y'$ (i.e., a 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 an incorrect statement is part of a new statement is not $1/2$ anymore 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 an incorrect statement. Note that the process of raising an incorrect statement to some exponent can also result in a correct statement. This is indeed very likely if an incorrect statement $x^{q^T} \stackrel{?}{=} y$ is "close" to the correct one in the sense that y is the correct result multiplied by a low-order element α. If, for example, this element α is of order two and the statement is raised to an even exponent, say two, the resulting statement $(x^{q^T})^2 \stackrel{?}{=} (y\alpha)^2$ will be a valid one. This observation underlies an attack on [40] that was first described[10] in [10] and it is also the reason why [40] 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 $x^{q^T} \stackrel{?}{=} y$, the (honest) prover only proves the (shorter) modified statement $x^{q^{T-C}} \stackrel{?}{=} y'$, where $y' := x^{q^{T-C}}$, and the verifier checks $(y')^{q^C} \stackrel{?}{=} y$ *by itself* as the final step. Moreover, to ensure that all the low orders are covered, we define q to be the

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

product of all small prime numbers up to a certain bound B as in Equation (1). Now, a malicious prover that tries to cheat on an original statement by proving a wrong modified statement[11] *will* get caught in the final exponentiation *as long as* the wrong modified statement is "close" to the correct one, where "close" means that the correct result 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 $x^{q^{T-C}} \stackrel{?}{=} y'\alpha$ (which is wrong), 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 adversaries that try to convince the verifier of statements that are "far" from correct, i.e., where the correct result 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 correct statements and the verifier falsely 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 [7], i.e., it is basically the number of parallel repetitions of Pietrzak's protocol.

We give a formal description of our protocol in Figure 3. For clarity of exposition[12], we assume that $T = 2^t + C$ for some $t \in \mathbb{N}$. Note that, similarly to [7], the starting instance in our protocol can either contain ρ many different statements with exponent q^T or ρ many copies of the same statement.

2.1 Soundness

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

Theorem 1. *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 $\mathcal{V}$ will output* `accept` *on an incorrect statement $(x, y, T = 2^t + C)$ with probability at most*

$$\frac{t}{B^\rho}.$$

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

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

[12] The case where $T - C$ is not a power of 2 can be handled by a standard approach similar to [40, Section 3.1].

Instance: (x, T, y), where $x, y \in \mathbb{G}$ and $T \in \mathbb{N}$

Parameters: (determined in the analysis)

1. bound $B \in \mathbb{N}$, which defines the base $q := \prod_{\text{prime } p<B} p$
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: $x^{q^T} = y$

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

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

Fig. 3. Our basic Proof of Exponentiation.

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 explain how the order of a group element affects soundness. Let $x^{q^{T-C}} = y'$ but a malicious prover claims that the result is $x^{q^{T-C}} = y'\alpha$. We say that the second statement is *α-wrong*. Then soundness of the protocol depends on the order of α:

In the execution of the protocol, the prover first sends a midpoint μ, which results in two statements $\mu \stackrel{?}{=} x^{q^{(T-C)/2}}$ and $\mu^{q^{(T-C)/2}} \stackrel{?}{=} y'\alpha$. Note that whatever the prover claims to be μ, one of the two statements will be incorrect, so for now we can assume that the prover sends a correct midpoint $\mu = x^{q^{(T-C)/2}}$. 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 correct 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 (for more information on the size of the randomness see Section 2.2). This means that whenever $\operatorname{ord}(\alpha)$ is large, it is unlikely that the statement is transformed into a correct statement after a single round. However, the order of the element that makes the statement incorrect can also decrease round by round until the statement is transformed into a correct one. To show this, we use the following well-known fact. A proof can be found in any standard textbook on group theory (e.g., [22, 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*

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

From Proposition 1 we get that $\operatorname{ord}(\alpha^{r_1+r_2+\dots+r_\rho}) < \operatorname{ord}(\alpha)$ whenever $r_1 + r_2 + \dots + r_\rho \equiv 0 \mod d$, where d is a divisor of $\operatorname{ord}(\alpha)$. If the order decreases in all of the ρ many new statements obtained this way, the adversary has a better chance to end up with a correct 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 correct. 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 wrong, gives ρ correct 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-wrong 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 instances $\{(\tilde{x}_j, \tilde{y}_j, T)\}_{j\in[1,\rho]}$ is $\tilde{\alpha}$-wrong 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 instances $\{(\tilde{x}_j, \tilde{y}_j, T)\}_{j\in[1,\rho]}$ is $\tilde{\alpha}$-wrong 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 results 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 $s \in \{0, 1, \ldots, e\}$. The probability that $s \geq k$ for any $k \in \{0, 1, \ldots, e\}$ 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 obtain the first claim of the lemma, we set $e = 1$ and $p = B$. The probability that the new statement is correct is the probability that $s = 1$, which is $1/B$. Hence, the probability that all of the ρ new instances are correct is $1/B^\rho$.

We obtain the second claim of the lemma by setting $e \geq \log(B)$ and observing that the probability of $s \geq \log(B)$ is $1/p^{\log(B)} \leq 1/2^{\log(B)} = 1/B$. Hence, 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 result in Step 2 of the protocol is $x^{q^{T-C}} = y'$ but a malicious prover claims that it is $x^{q^{T-C}} = y'\alpha$ (i.e., makes a statement that is α-wrong). Notice that in the case where $\mathrm{ord}(\alpha) \mid q^C$ we have that $(y'\alpha)^{q^C} = (y')^{q^C} = y$ and, hence, the verifier ends up rejecting after Step 3 of the protocol. It follows that an adversary who wants to convince the verifier that the result is not y needs to choose an element α of order not dividing q^C. The adversary wins if all of the ρ statements are correct after t rounds of the protocol. From the discussion above we know that the best option for the adversary 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 analyze the two cases separately.

Case 1: Let $\mathrm{ord}(\alpha) = p$. Assume that in round i of the protocol we have ρ many statements $\{(x_{i-1,j}, y_{i-1,j}\alpha^{a_{i-1,j}}, T_{i-1})\}_{j\in[1,\rho]}$ 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 correct. Otherwise it is wrong 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 correct in round $i+1$.
In Step 2 of the protocol, the prover sends midpoints $\mu_{i,j}$ which results in 2ρ statements

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

which we denote by $\{(u_{i,k}, v_{i,k}\alpha^{b_{i,k}}, T_i)\}_{k\in[1,2\rho]}$. Note that at least one of the $b_{i,k}$ is nonzero 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 correct is at most $1/B^\rho$. By the union bound, we get that the probability that all statements are correct after t rounds is

$$\frac{t}{B^\rho}.$$

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 correct statement after t rounds, the adversary has to decrease the order of the wrong 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 of the form $\{(x_{i-1,j}, y_{i-1,j}\alpha^{a_{i-1,j}}, T_{i-1})\}_{j\in[1,\rho]}$ where $a_{i-1,j} \in \mathbb{Z}$. 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

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

which we denote by $\{(u_{i,k}, v_{i,k}\alpha^{b_{i,k}}, T_i)\}_{k\in[1,2\rho]}$.

We note that whatever midpoint the prover sends, the order of the element that makes one of the two statements $\mu_{i,1} \stackrel{?}{=} x_{i-1,1}^{q^{T_i}}$ and $\mu_{i,1}^{q^{T_i}} \stackrel{?}{=} y_{i-1,1}\alpha^{a_{i-1,1}}$ incorrect is at least $\mathrm{ord}(\alpha^{a_{i-1,1}})$. To see this, assume that $\mu_{i,1}$ is the correct midpoint but the adversary sends $\mu_{i,1}\beta$ for some group element β. Then the second statement becomes $\mu_{i,1}^{q^{T_i}} \stackrel{?}{=} y_{i-1,1}\alpha^{a_{i-1,1}}\beta^{-q^{T_i}}$, which is γ-wrong 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 $\tilde{\alpha}$-wrong, 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 the adversary wins after t rounds with probability at most

$$\frac{t}{B^\rho}.$$

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.*

Proof. As we have seen above, a malicious prover is able to convince the verifier of a wrong 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-wrong statement where $\mathrm{ord}(\alpha)$ has a prime divisor of size at least B is transformed into a correct one or
- the order of the wrong 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 the union bound, the probability that a malicious prover that makes up to Q queries to the random oracle will find such a query is at most $Q \cdot (1/B)^\rho$. □

2.2 Efficiency

In this section, we analyze the efficiency of the Fiat-Shamir transform of our PoE for proving a statement of the form $x^{q^T} \stackrel{?}{=} y$ 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

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

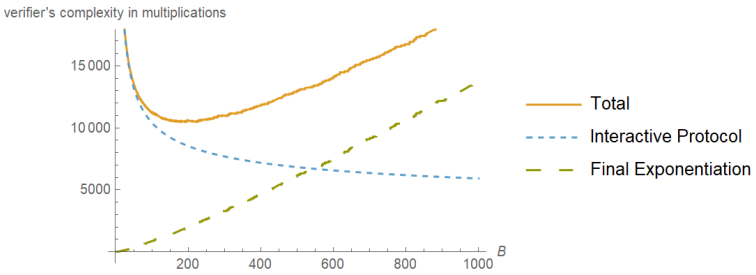

Fig. 4. Number of multiplications of the verifier in one round for 80-bit security depending on the bound B. The orange graph is the total verifier's complexity for one round, the blue dotted graph is the cost of the interactive part of the protocol and the green dashed graph is the cost of the final exponentiation divided by the number of rounds (i.e., we amortize the cost of the final exponentiation over the number of rounds).

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 "square and multiply" 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 [24] . 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 4, 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 [7], 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

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

Table 1. Comparison of different PoEs. Verifier's complexity is measured in the number of multiplications and proof-size $|\pi|$ in the number of group elements.We denote by λ the statistical security parameter. [40] is statistically-sound only in groups without elements of small order.

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 [40, 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 1, we compare our protocol with the proofs of exponentiation from [40], [7], and [50]. 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 [50] is the most efficient PoE regarding verifier's complexity and proof-size. However, it is not statistically-sound. [40] 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 [7] 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 [7] and we compare the communication complexity per round for different values of B in Figure 1.

The verifier's efficiency of our PoE depends on the choice of the bound B which also determines the size of q. In Figure 2, we compare the number of

multiplications per round for the verifier in both protocols for different choices of B. Additionally to the work in each round, the verifier computes λ many exponentiations with q in the last round of [7] 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 [7] 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 exponentation 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-) 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 [7], 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 [43] 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 [40] and [7].

3.1 The Protocol

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

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

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

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

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 of the form

$$u_j^{q^{2^s}} \stackrel{?}{=} v_j$$

for $j \in [2\rho]$. The prover makes this $2\rho+1$ statements by adding $g_2^{q^{2^s}} \stackrel{?}{=} h_2'$ to these statements. 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 incorrect. In the end the verifier checks if $(h_1')^{q^{C_1}} = h_1$ and $(h_2')^{q^{C_2}} = h_2$. This process can be extended to arbitrary-many statements of the form $g_i^{q^{2^r}+C_i} \stackrel{?}{=} h_i$ with the protocol given in Figure 5. Note that in Step 4 we do not specify whether the verifier checks $(h_i')^{q_i^C} = h_i$ by carrying out the computation locally or by appending the statement to the instances. 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 $x^{q^S} \stackrel{?}{=} y$ 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 $x^{q^{s_0}} \stackrel{?}{=} y_1$ and $y_i^{q^{s_i \cdot 2^i}} \stackrel{?}{=} y_{i+1}$ 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:*

1. *$\mathcal{P}$ and $\mathcal{V}$ 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. *$\mathcal{V}$ 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.

Instance: $\left\{(g_i, h_i, 2^{t_i} + C_i)\right\}_{i\in[1,m]}$ with $g_i, h_i \in \mathbb{G}$ and $t_1 > t_2 > \ldots > t_m \in \mathbb{N}$

Claim: $g_i^{q^{2^{t_i}+C_i}} = h_i$ over $\mathbb{G}$ for all $i \in [1, m]$ and $q \in \mathbb{N}$

Parameters: (determined in the analysis)

1. number of rounds of parallel repetition ρ
2. size of individual random coin κ

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 instances of the form $\left\{(x_j, y_j, 2^{t_{i+1}+1})\right\}_{j\in[1,\rho]}$:
   (a) The prover sends ρ midpoints $\{\mu_j\}_{j\in[1,\rho]}$, which results in 2ρ instances $\left\{(u_k, v_k, 2^{t_{i+1}})\right\}_{k\in[1,2\rho]}$
   (b) The prover and verifier append $(g_{j+1}, h'_{j+1}, 2^{t_{i+1}})$ to the instances resulting in $2\rho + 1$ instances of the form $\left\{(\tilde{u}_k, \tilde{v}_k, 2^{t_{i+1}})\right\}_{k\in[1,2\rho+1]}$.
   (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 $\left\{(\tilde{x}_j, \tilde{y}_j, 2^{t_{i+1}})\right\}_{j\in[1,\rho]}$ as the instance 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 $\left\{(x^*_j, y^*_j, 1)\right\}_{j\in[1,\rho]}$.
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 $i \in [1, m]$.

Fig. 5. Batching protocol for PoE.

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 $\mathcal{V}$ will output* `accept` *on instance $\{(g_i, h_i, 2^{t_i}+C_i)\}_{i\in[1,m]}$, where $t_1 \geq t_2 \geq \ldots \geq t_m$ and at least one statements is incorrect, with probability at most*

$$\frac{t_1}{B^\rho}.$$

3.2 Improving Verifier's Efficiency

In this section we analyze how the batching protocol reduces the number of multiplications for verifying a statement of the form $x^{q^T} \stackrel{?}{=} y$. In Appendix B.1 we analyze the gain in efficiency of the polynomial commitment in [7] when we use this improved version of our PoE as a building block instead of the PoE proposed in [7].

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 instances

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

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

References

1. H. Abusalah, C. Kamath, K. Klein, K. Pietrzak, and M. Walter. Reversible proofs of sequential work. In Y. Ishai and V. 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.

2. A. Arun, C. Ganesh, S. Lokam, T. Mopuri, and S. Sridhar. Dew: Transparent constant-sized zkSNARKs. Cryptology ePrint Archive, Paper 2022/419, 2022. https://eprint.iacr.org/2022/419.
3. K. Belabas, T. Kleinjung, A. Sanso, and B. 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. M. Bellare, J. A. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In K. 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.
5. M. J. Beller and Y. Yacobi. Batch Diffie-Hellman key agreement systems and their application to portable communications. In R. A. Rueppel, editor, *Advances in Cryptology — EUROCRYPT' 92*, pages 208–220, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.
6. N. Bitansky, S. Goldwasser, A. Jain, O. Paneth, V. Vaikuntanathan, and B. Waters. Time-lock puzzles from randomized encodings. In M. 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.
7. A. R. Block, J. Holmgren, A. Rosen, R. D. Rothblum, and P. Soni. Time- and space-efficient arguments from groups of unknown order. In T. Malkin and C. 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.
8. J. Blocki, S. Lee, and S. Zhou. On the security of proofs of sequential work in a post-quantum world. In S. 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.
9. D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In H. Shacham and A. 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.
10. D. Boneh, B. Bünz, and B. Fisch. A survey of two verifiable delay functions. *IACR Cryptol. ePrint Arch.*, 2018:712, 2018.
11. D. Boneh and M. Naor. Timed commitments. In M. 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.
12. C. Boyd and C. 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. J. Buchmann and H. C. Williams. A key-exchange system based on imaginary quadratic fields. *J. Cryptol.*, 1(2):107–118, 1988.
14. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In A. Canteaut and Y. 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.
15. 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.
16. J. Chavez-Saab, F. R. Henríquez, and M. Tibouchi. Verifiable isogeny walks: Towards an isogeny-based postquantum VDF. Cryptology ePrint Archive, Report 2021/1289, 2021. `https://ia.cr/2021/1289`.
17. M. Chen, R. Cohen, J. Doerner, Y. Kondi, E. Lee, S. Rosefield, and A. Shelat. Multiparty generation of an RSA modulus. In D. Micciancio and T. 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.
18. A. R. Choudhuri, P. Hubáček, C. Kamath, K. Pietrzak, A. Rosen, and G. N. Rothblum. PPAD-hardness via iterated squaring modulo a composite. Cryptology ePrint Archive, Report 2019/667, 2019. `https://ia.cr/2019/667`.
19. B. Cohen and K. Pietrzak. Simple proofs of sequential work. In J. B. Nielsen and V. 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.
20. N. Döttling, S. Garg, G. Malavolta, and P. N. Vasudevan. Tight verifiable delay functions. In C. Galdi and V. 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.
21. N. Döttling, R. W. F. Lai, and G. Malavolta. Incremental proofs of sequential work. In Y. Ishai and V. 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.
22. D. S. Dummit and R. M. Foote. *Abstract Algebra*. John Wiley and Sons, 3rd edition, 2003.
23. N. Ephraim, C. Freitag, I. Komargodski, and R. Pass. Continuous verifiable delay functions. In A. Canteaut and Y. 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.
24. P. 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.
25. L. D. Feo, S. Masson, C. Petit, and A. Sanso. Verifiable delay functions from supersingular isogenies and pairings. In S. D. Galbraith and S. 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.
26. A. Fiat. Batch RSA. *J. Cryptol.*, 10(2):75–88, 1997.

27. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1986.
28. T. K. Frederiksen, Y. Lindell, V. Osheter, and B. Pinkas. Fast distributed RSA key generation for semi-honest and malicious adversaries. In H. Shacham and A. 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.
29. C. Freitag and I. 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`.
30. C. Freitag, I. Komargodski, R. Pass, and N. Sirkin. Non-malleable time-lock puzzles and applications. In K. Nissim and B. 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.
31. C. Hoffmann, P. Hubáček, C. Kamath, K. Klein, and K. Pietrzak. Practical statistically-sound proofs of exponentiation in any group. Cryptology ePrint Archive, Report 2022/???, 2022.
32. J. Katz, J. Loss, and J. Xu. On the security of time-lock puzzles and timed commitments. In R. Pass and K. 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.
33. A. K. Lenstra and B. Wesolowski. Trustworthy public randomness with sloth, unicorn, and trx. *Int. J. Appl. Cryptogr.*, 3(4):330–343, 2017.
34. A. Lombardi and V. Vaikuntanathan. Fiat-Shamir for repeated squaring with applications to PPAD-hardness and VDFs. In D. Micciancio and T. 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.
35. M. Mahmoody, T. Moran, and S. P. Vadhan. Time-lock puzzles in the random oracle model. In P. 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.
36. M. Mahmoody, T. Moran, and S. P. Vadhan. Publicly verifiable proofs of sequential work. In R. D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 373–388. ACM, 2013.
37. M. Mahmoody, C. Smith, and D. 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.
38. T. C. May. Timed-release crypto, 1994.
39. D. M'Raïhi and D. Naccache. Batch exponentiation: A fast DLP-based signature generation strategy. In L. Gong and J. 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.
40. K. Pietrzak. Simple verifiable delay functions. In A. 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.
41. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
42. 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.
43. L. Rotem. Simple and efficient batch verification techniques for verifiable delay functions. In K. Nissim and B. 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.
44. L. Rotem and G. Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In D. Micciancio and T. 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.
45. L. Rotem, G. Segev, and I. Shahaf. Generic-group delay functions require hidden-order groups. In A. Canteaut and Y. 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.
46. P. Schindler, A. Judmayer, M. Hittmeir, N. Stifter, and E. 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.
47. B. Shani. A note on isogeny-based hybrid verifiable delay functions. Cryptology ePrint Archive, Report 2019/205, 2019. `https://ia.cr/2019/205`.
48. P. Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In R. 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.
49. A. van Baarsen and M. 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.
50. B. Wesolowski. Efficient verifiable delay functions. *J. Cryptol.*, 33:2113–2147, 2020.
51. B. Wesolowski and R. Williams. Lower bounds for the depth of modular squaring. Cryptology ePrint Archive, Report 2020/1461, 2020. `https://ia.cr/2020/1461`.

A Improving Verifier's efficiency

In Figure 2 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

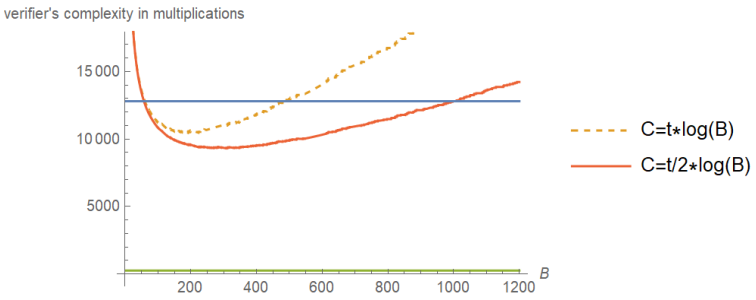

Fig. 6. Number of multiplications of the verifier in one round for 80-bit security depending on the bound B. The blue line is the number of multiplications in [7], the dotted orange graph is the complexity of our protocol with $C = t\log(B)$, the red graph is the complexity in our protocol with $C = t\log(B)/2$ and the green line is the verifier's complexity in [40].

reduce this number to $C = t\log(B)/2$ by setting q to

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

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 6.

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 7. We conclude that the verifier's complexity of our scheme improves upon [7] 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 analyse the gain in efficiency when we use our PoE as a building block instead of the one proposed in [7].

In the full version of the paper [31] we provide an overview of the polynomial commitment scheme in [7]. Here we only state the key properties that the PoE should satisfy in order to be applicable in the polynomial commitment scheme.

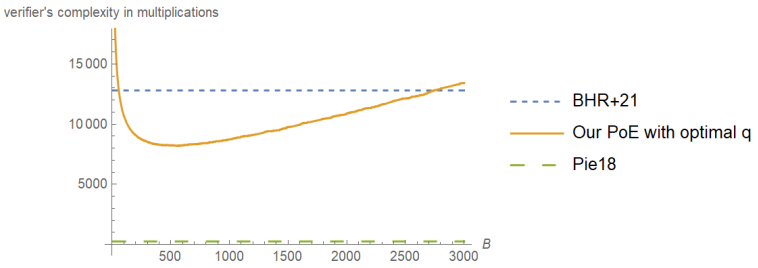

Fig. 7. Number of multiplications of the verifier in one round for 80-bit security depending on the bound B. The dotted blue line is the number of multiplications in [7], the orange graph is the complexity of our protocol with $C = t$ and q as above and the green line is verifier's complexity in [40] (which is 240 multiplications).

Requirements from the PoE. Note that the use of the PoE in the [7] 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 ([7, Lemma 6.4]).[13] Our PoE satisfies statistical soundness.
2. Secondly, the base q used in the PoE protocol is borrowed *from* the polynomial commitment. In order for the polynomial commitment to satisfy its homomorphic properties, [7] 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 (1). However, we show in the full version of the paper [31] that the requirement that q be odd is not necessary in [7].

B.1 Efficiency

In this section we analyze the improvement in efficiency of the polynomial commitment scheme in [7] 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 [7] the communication complexity of proving λ many statements with the same exponent is $\lambda(n-k-1)$ group elements. This

[13] 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.

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 [7] 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 [7].