Practical Batch Proofs of Exponentiation

Charlotte Hoffmann
charlotte.hoffmann@ist.ac.at
Institute of Science and Technology Austria
Klosterneuburg, Austria

Pavel Hubáček
hubacek@math.cas.cz
Institute of Mathematics, Czech Academy of Sciences
Charles University, Faculty of Mathematics and Physics
Prague, Czech Republic

Svetlana Ivanova
ivanovsv@o365.cuni.cz
Charles University, Faculty of Mathematics and Physics
Prague, Czech Republic

ABSTRACT

A Proof of Exponentiation (PoE) allows a prover to efficiently convince a verifier that $y = x^e$ in some group of unknown order. PoEs are the basis for practical constructions of Verifiable Delay Functions (VDFs), which, in turn, are important for various higher-level protocols in distributed computing. In applications such as distributed consensus, many PoEs are generated regularly, motivating protocols for secure aggregation of batches of statements into a few statements to improve the efficiency for both parties. Rotem (TCC 2021) recently presented two such generic batch PoEs.

In this work, we introduce two batch PoEs that outperform both proposals of Rotem and we evaluate their practicality. First, we show that the two batch PoEs of Rotem can be combined to improve the overall efficiency by at least a factor of two. Second, we revisit the work of Bellare, Garay, and Rabin (EUROCRYPT 1998) on batch verification of digital signatures and show that, under the low order assumption, their bucket test can be securely adapted to the setting of groups of unknown order. The resulting batch PoE quickly outperforms the state of the art in the expected number of group multiplications with the growing number of instances, and it decreases the cost of batching by an order of magnitude already for hundreds of thousands of instances. Importantly, it is the first batch PoE that significantly decreases both the proof size and complexity of verification. Our experimental evaluations show that even a non-optimized implementation achieves such improvements, which would match the demands of real-life systems requiring large-scale PoE processing.

Finally, even though our proof techniques are conceptually similar to Rotem, we give an improved analysis of the application of the low order assumption towards secure batching of PoE instances, resulting in a tight reduction, which is important when setting the security parameter in practice.

CCS CONCEPTS

• **Security and privacy → Mathematical foundations of cryptography**; **Distributed systems security**; **Public key (asymmetric) techniques**.

KEYWORDS

Proof of Exponentiation, Verifiable Delay Function, Batching

1 INTRODUCTION

In recent years, various "time-based" cryptographic schemes were constructed relying on verifiable delay functions – functions assumed to be sequentially hard, i.e., that they cannot be computed faster than a prescribed amount of time, while their output can be efficiently verified given a succinct proof. Example applications include the design of blockchains [14], randomness beacons [34, 37], proofs of data replication [8], computational time-stamping [13, 29] and short-lived zero-knowledge proofs and signatures [1]. Although the constructions and applications of such schemes are numerous, the known sources of sequential hardness are still relatively scarce. The core computational hardness assumption is the heuristic assumption about the inherent sequentiality of repeated squaring in groups of unknown order due to Rivest, Shamir, and Wagner [35], i.e., that computing x^{2^T} in a group of unknown order cannot be significantly sped up below T group operations even leveraging parallelism.

Pietrzak [32] and Wesolowski [43] recently demonstrated the first protocols for proving that a repeated squaring instance has been computed correctly even without the knowledge of the group order. Such *Proofs of Exponentiation* (PoEs) consist of few group elements, and verifying their correctness is much faster than recomputing the repeated squaring instance – usually logarithmic in the number of squarings required to compute the result.

One practically important feature of PoEs is that many statements can be efficiently aggregated to a single statement. In particular, m statements $y_1 = x_1^e, \ldots, y_m = x_m^e$ can be represented by a single aggregate statement $\prod_{i=1}^m y_i = \left(\prod_{i=1}^m x_i\right)^e$. This property suggests that one might be able to construct *batch* Proofs of Exponentiation, i.e., secure aggregation protocols that allow one to verify many statements by checking a single PoE. Clearly, the above naive aggregation does not have any meaningful security properties – it is easy to produce incorrect statements that would be aggregated into a valid statement, e.g., by randomly permuting the correct y_i's. However, a sound batching procedure can be constructed by randomizing the aggregation using a suitable vector of exponents $r_1, \ldots, r_m \in \mathbb{Z}_p$ and considering the batched statement $\prod_{i=1}^m y_i^{r_i} = \left(\prod_{i=1}^m x_i^{r_i}\right)^e$.

Notice that, in the extreme case where the randomization exponents are uniformly random bits $r_1, \ldots, r_m \in \mathbb{Z}_2$, the procedure amounts to the naive aggregation performed on a random subset of the original statements. However, it already achieves soundness error $\frac{1}{2}$ for an arbitrary collection of statements.[1] There are two natural approaches for boosting the soundness (i.e., reducing the

[1] This can be seen by the following argument: Suppose that the i-th statement is false and batch all statements except for the i-th one. Denote by b the uniformly random bit determining whether the i-th statement is multiplied to the batched statement or not. If the batched statement is correct, then it will remain correct if and only if $b = 0$. If it is wrong, one needs at least that $b = 1$ for it to become correct. In both cases, the probability of resulting in a correct statement is at most $\frac{1}{2}$.

soundness error). In applications requiring statistical security, λ parallel instances of the random subset approach can be run to get soundness error $2^{-\lambda}$. We refer to the parallel repetition variant of the random subset approach as the *Random Subsets Protocol*. It was independently suggested by Block et al. [7] and Rotem [36].

The overhead stemming from the λ parallel repetitions can be avoided when one is willing to settle for computational soundness guarantees from the batching process. In particular, one can perform a single randomized batching using exponents from a sufficiently large set, which is computationally sound, assuming that the adversary cannot efficiently produce group elements of low order. The corresponding batch PoE, which we call the *Random Exponents Protocol*, was proposed and analysed by Rotem [36].

The practical efficiency of the resulting batch PoE ultimately depends on the number of group multiplications performed. When raising the instances to uniformly random ℓ-bit exponents before computing their product, one has to perform 1.5ℓ multiplications in expectation per each exponentiation. Thus, for $\ell = \lambda$, the Random Subsets Protocol based on computing products of λ random subsets results (in expectation) roughly in half as much work as in the Random Exponents Protocol that raises each statement to a random λ exponent before computing their product. However, when batching via the Random Subsets Protocol, λ proofs have to be constructed and verified, while the Random Exponents Protocol results in a single PoE. Note that proofs can be efficiently verified, but the construction is not as efficient, so proving λ many statements instead of one yields a significant loss in the prover's efficiency in practice. Hence, the above two batch PoEs from the literature give a trade-off for the complexities of the verifier and prover.

To evaluate the improvement achieved by the above batch PoEs, we need to compare them to the baseline when one disregards any attempt at batching and simply verifies all PoEs, which, e.g., avoids any implementation overhead of batching. Clearly, batch PoEs minimize the storage requirement, as instead of storing m proofs, one can store λ or even a single proof. Below, we discuss the number of expected multiplications per instance in the PoEs from Pietrzak [32] and Wesolowski [43] to compare with the verification complexity in the known batch PoEs.

Verification cost of Wesolowski's PoEs. Wesolowski's proof contains a single group element π, and the verifier needs to verify a single group identity $y = \pi^s x^r$, where s and r are ℓ-bit integers. This can be done using at most $3\ell + 1$ multiplications in expectation. Thus, if we set ℓ to the security parameter λ, then the expected multiplication overhead of verifying proofs for all m instances is roughly the same as in the Random Exponents Protocol. However, for the non-interactive variant of Wesolowski's protocol, it is necessary to set $\ell = 2\lambda$ due to an attack by Boneh, Bünz, and Fish [9] Therefore, the Random Exponents Protocol is likely to speed up the batch verification process by a factor of two in practice.[2]

Verification cost of Pietrzak's PoEs. Pietrzak's proof is based on an interactive folding approach that proceeds in $\log T$ rounds for statements of the form $y = x^{2^T}$. Given x_i and y_i constructed in round i, the verifier needs to construct a new statement with $x_{i+1} = x_i^r \cdot \mu$ and $y_{i+1} = \mu^r \cdot y_i$, where r is an ℓ-bit integer. This can be perfomed using at most $3\ell + 2$ multiplications in expectation. In the final round, the verifier checks a group identity of the form $\tilde{y} = \tilde{x}^2$ using a single multiplication. Thus, the overall multiplication cost can be estimated as $(3\ell + 2) \log T$. Due to the dependency on the time delay parameter T, the effects of batching would be more pronounced for Pietrzak's PoE.

Practical importance of batch PoEs. In higher-level protocols, many PoE proofs are commonly generated, and there is a clear incentive to batch them to minimize the communication complexity as well as the computational overhead for the users. For example, PoE-based VDFs can be used in the design of distributed consensus protocols together with "time-less" primitives such as Proof of Space [15]. Consider the Chia Network [26] blockchain that currently generates 32 VDF proofs in expectation every 10 minutes and has a block length of around 4.8 million blocks. One possible application of batch PoEs for such blockchains is for efficient onboarding of new users with "light" clients. Specifically, miners could provide a single batch PoE for all intermediate PoE statements. Thus, the storage overhead and clients' computation are reduced simultaneously.

1.1 Our Contributions

In this work, we present two new approaches for batching PoEs that improve over the above protocols in various ways.

The Hybrid Protocol. The first approach, which we call the *Hybrid Protocol*, is a natural combination of the two approaches above. It only slightly increases the work performed by the verifier compared to the Random Subsets Protocol but results in a single PoE proof that has to be constructed and verified. Specifically, using the Random Subsets Protocol described above, m instances are first batched to λ instances, which are then batched to a single PoE instance via the Random Exponents approach with λ-bit exponents. Our experiments show that this new batch PoE, while very easy to implement, is already roughly twice as fast as the Random Exponents Protocol.

The Bucket Protocol. The second approach, which we call the *Bucket Protocol*, can be seen as an optimization of the Hybrid Protocol that, for a large enough number of instances, is one order of magnitude faster than the random exponents approach and five times faster than the random subsets approach. It is based on a protocol of Bellare, Garay and Rabin [6] for batch verification of signatures in prime-order groups. The main difference to our construction is that we ultimately aggregate all statements into one, such that only one proof needs to be computed and verified in the end, while the original batching of Bellare et al. yields multiple proofs that need to be verified in parallel.

The main idea of the Bucket Protocol is to optimize the number of multiplications incurred in the Random Subsets Protocol by decreasing the number of parallel repetitions as follows: In the Random Subsets Protocol, a new PoE statement is constructed by taking a random subset of the m original PoE statements, which is of expected size half the number of instances, and computing the product of the statements in the subset. Given m instances, repeating this step λ times to amplify the soundness results in

[2] There might be additional gains since, when verifying the Wesolowski's proof, the verifier needs to additionally perform some computation, which induces non-trivial overhead. Attias et al. [3] reported that, for most parameters, the additional overhead might amount to at least half of the verification time (see Table 1 in [3]).

$\lambda m/2$ multiplications in expectation. In the Bucket Protocol, each of the m instances is assigned uniformly at random to one of $K = 2^k$ buckets, where k is a parameter of the protocol. Each bucket then gives rise to a new statement by taking the product of the statements that are assigned to that bucket, and the resulting K instances are then batched via the Random Exponents Protocol using random exponents of size k. The soundness analysis shows that, in this case, $\rho \approx \lambda/k$ parallel repetitions are sufficient, reducing the number of statement multiplications from $\lambda m/2$ to ρm, whereas the number of exponentiations is independent of m. In particular, for any specific number of statements m, we can find the best k that minimizes the number of group multiplications. We discuss the optimal choice of k and its effect on performance in Section 5.

The soundness analysis. Our soundness analysis of the Bucket Protocol differs from Bellare et al. as they considered the setting of prime-order groups, whereas we analyze the protocol in groups of hidden order. The main tool we use in the soundness analysis of our protocols is a technical lemma (Lemma 2) which (informally) states that an adversary can only win the following game with probability at most 1 over the size of the randomness space or by breaking the low order assumption: Output a set of statements, of which at least one is incorrect, and which yield a correct statement after raising them to random exponents and multiplying them. The intuition behind the proof is the following: If the statement $y \stackrel{?}{=} x^e$ is incorrect in a group, then you can think of the group element y as being obtained by multiplying the correct result by another "bad" element. This holds without loss of generality, because every element has an inverse in a group. Now, if you raise the statement to a random exponent r, then the bad element vanishes from the equation if and only if r is a multiple of its order, which happens with probability approximately 1 over the order. Rotem proves a similar lemma in [36]. However, his reduction from the low order assumption incurs a quadratic loss in the winning probability because it guesses the order of the low order element it outputs. We observe that, whenever the winning probability of the adversary is not negligible, the reduction can efficiently find the order of the element it outputs and, hence, it has the same winning probability as the adversary. Furthermore, the lemma in [36] depends on the bound on the order of low-order elements in the assumption, i.e., the bound that defines when an element is considered of "low-order". We remove this dependency as it is not needed for the analysis, and thereby clarify the role of the different parameters.

Overview of all protocols. In Figure 1, we visualise the various batching approaches. All protocols run on m PoE instances represented by the black squares.

- **Random Subsets Protocol (Figure 1a):** The prover computes the product of λ independent uniformly random subsets of the instances sampled by the verifier and outputs PoE proofs for all λ resulting PoE instances represented by the gray squares.
- **Random Exponents Protocol (Figure 1b):** The prover computes the product of the original instances after raising them to uniformly random λ-bit exponents. The thicker arrows compared to the first subfigure correspond to the higher cost of the exponentiation in terms of group operations (i.e., 1.5λ group multiplications in expectation).
- **Hybrid Protocol (Figure 1c):** The prover uses the Random Exponents Protocol to batch the λ statements produced by the Random Subsets Protocol. Crucially, the more costly exponentiation to λ-bit uniformly random exponents is always performed only to λ statements independently of the total number of instances m.
- **Bucket Protocol (Figure 1d):** First, there are only $\rho \approx \lambda/k$ parallel repetitions of the step when the initial m statements are placed in the 2^k buckets. Second, the batched instances corresponding to each bucket are batched using "small" k-bit uniformly random exponents. Finally, the most costly Random Exponents Protocol is run to batch only ρ instances.

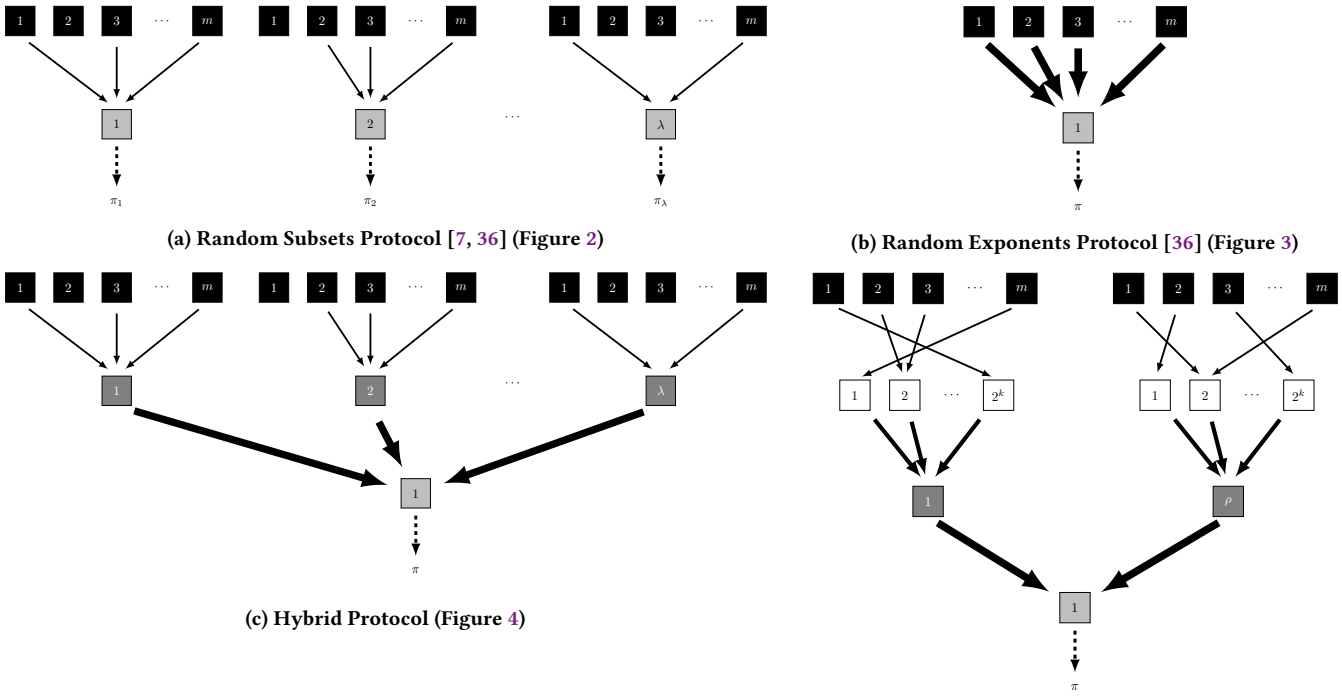
1.2 Related Work

Batch verification of digital signatures. A related line of work initiated by Fiat [18] considers the batch verification of many digital signatures. See Camenish, Hohenberger, and Pedersen [10] for a historical overview of this line of work. Bellare, Garay, and Rabin [6] presented a set of protocols for batch verification of modular exponentiation in prime-order groups. For a generator g of a group $\mathbb{G}$, they consider the problem of verifying statements of the form $y_1 = g^{e_1}, \ldots, y_m = g^{e_m}$. Importantly, they work in groups with a publicly known order, and wish to batch verify exponentiations with a fixed base and varying exponents. However, their protocols can also be adapted to the relevant setting relevant for VDFs. Di Crescenzo et al. [16] extended the small exponent protocol of [6] to the setting of safe prime RSA groups. However, their proof size is linear in the number of statements to be batched.

Efficient multi-exponentiation. Group multi-exponentiation, i.e., evaluation of products of the form $\prod_{i=1}^{k} g_i^{e_i}$, has long been recognized as an important algorithmic task in cryptography. Going back to Straus [41] and Pippenger [33], various algorithms for multi-exponentiation were proposed (see Möller [31] and Henry [21] and the references therein). Attias et al. [4] recently presented an experimental evaluation of the known multi-exponentiation algorithms on various architectures. We discuss the effects of more efficient multi-exponentiation algorithms on our protocols in Section 5.1.

Constructions of PoEs. Even though proofs of exponentiation were introduced only recently, there are already several constructions for different applications. The first two PoEs were introduced concurrently by Wesolowski [43] and Pietrzak [32] in the context of VDFs. Block et al. [7] presented the first statistically-sound PoE in any group, which is needed as a building block in a polynomial commitment. The PoE in [23] is built on their protocol and reduces the complexity for a special choice of exponent. The PoE in [24] works in an extension field of $\mathbb{Z}_N$ and is used to build a VDF where the delay property relies on a potentially weaker assumption than iterated squaring. In [25] a PoE for Proth number groups is presented to certify proofs of non-primality to improve the efficiency of large-scale distributed projects for computational number theory.

Batching of PoEs. The PoEs in [7, 23] achieve statistical soundness via a parallel repetition of Pietrzak's PoE and can be easily



(a) Random Subsets Protocol [7, 36] (Figure 2)

(b) Random Exponents Protocol [36] (Figure 3)

(c) Hybrid Protocol (Figure 4)

(d) Bucket Protocol (Figure 5)

Figure 1: Depiction of the known (top) and new (bottom) batch Proofs of Exponentiation. Squares represent PoE instances. The thickness of full arrows highlights the cost in group operations, i.e., thicker arrows correspond to less efficient aggregation. Dashed arrows represent computation of the final PoE(s).

modified to prove arbitrarily many statements at once, resulting in the Hybrid Protocol. Rotem [36] concurrently presented a similar batching technique as [7] and the Random Exponents Protocol. His protocols can be viewed as adaptations of two protocols in [6] to groups in which the low order assumption holds. Similarly, the most efficient protocol in our work is an adaptation of a protocol in [6] to those groups. The main difference is that the protocols in [6] are constructed for and analyzed in prime-order groups. We note that the batching protocol in [23] is the only one that works for statements with different exponents, but it specifically uses Pietrzak's PoE, whereas the other batching protocols are more flexible as to which PoE is run after aggregating the statements. Therefore, constructing a general batching protocol that handles different exponents remains an open problem.

Other VDFs. There are several candidate VDFs with sequentiality not based on iterated squaring, such as the permutation-polynomial based construction [8], the isogenies-based constructions [11, 17, 40] and the constructions from lattice problems [12, 28]. The constructions in [17, 40] work in the algebraic setting of isogenies of elliptic curves. Although these constructions provide a certain form of quantum resistance, they are presently far from efficient. Freitag et al. [20] constructed VDFs from any sequentially hard function and polynomial hardness of learning with errors, the first from standard assumptions. The works of Cini, Lai, and Malavolta [12, 28] constructed the first VDF from lattice-based assumptions and conjectured it to be post-quantum secure. Finally, some VDF candidates rely on "arithmetization friendly" symmetric primitives and practically efficient SNARKs [8, 27, 38]. However, the sequentiality of SNARK-based VDFs is yet to be understood, as shown, e.g., by the recent analysis of MinRoot [30].

Assumptions in hidden order groups. The low order assumption in RSA groups is analyzed in [39]. The authors give equivalence results for weaker forms of the low order assumption and the factoring assumption for a non-negligible portion of moduli. The low order assumption in class groups is analyzed in [5]. The authors show that it is broken for Mersenne primes and other special forms of prime numbers.

Performance of PoEs in Practice. Attias, Vigneri, and Dimitrov [2] reported on the practical performance of verification for two PoEs in the context of Verifiable Delay Functions in RSA groups. The verification time for Wesolowski's VDF can be further optimized in RSA groups, e.g., as suggested by Attias et al. [3].

2 PRELIMINARIES

In the rest of the paper, we let λ denote a security parameter. We use $[n] := \{1, \ldots, n\}$ to denote the set of all positive integers smaller than or equal to n.

Low order assumption. The following assumption was first formalized in [9]. It states that, in certain groups, it is (computationally) hard to find elements of low order.

Definition 1 (low order assumption)**.** Let $\mathsf{GGen}(1^\lambda)$ be a randomized algorithm that outputs the description of a group $\mathbb{G}$ of unknown order. We say that the *low order assumption* holds for GGen if, for any probabilistic polynomial-time algorithm $\mathcal{A}$, the probability of winning the following game is negligible in λ:

(1) $\mathcal{A}$ takes as input the description of a group $\mathbb{G}$ output by $\mathsf{GGen}(1^\lambda)$.
(2) $\mathcal{A}$ outputs a pair $(d, \alpha) \in [2^\lambda] \times \mathbb{G}$.
(3) $\mathcal{A}$ wins if and only if $\alpha \neq 1$ and $\alpha^d = 1$.

Multiplication cost of group exponentiation. A commonly used technique for group exponentiation is the square-and-multiply algorithm. The multiplication cost of computing x^e is at most $\lfloor \log(n) \rfloor + w(e)\lfloor \log(n) \rfloor$, where n is the bit length of e and $w(e)$ its Hamming weight (see, e.g., [42]). Therefore, when raising group elements to exponents of bit length n, we will use $1.5\log(n)$ as an upper bound on the expected number of multiplications.

Interactive proofs. For the formal analysis, we rely on the following standard notions from the context of interactive proof systems.

Definition 2 (interactive proof)**.** For a function $\varepsilon : \mathbb{N} \to [0, 1]$, an *interactive proof for a language* L is a pair of interacting algorithms $(\mathcal{P}, \mathcal{V})$, where $\mathcal{V}$ is a PPT algorithm, which satisfies the following properties:

- **Completeness:** For every $x \in L$, if $\mathcal{V}$ interacts with $\mathcal{P}$ on the common input x, then $\mathcal{V}$ accepts with probability 1.
- **Soundness:** For every $x \notin L$ and every (computationally unbounded) cheating prover strategy $\widetilde{\mathcal{P}}$, the verifier $\mathcal{V}$ accepts when interacting with $\widetilde{\mathcal{P}}$ with probability less than $\varepsilon(|x|)$, where ε is called the *soundness error*.

Definition 3 (proof of exponentiation)**.** A *proof of exponentiation* (PoE) in a group $\mathbb{G}$ is an interactive proof for the language

$$L = \{(x, y, e) \in \mathbb{G}^2 \times \mathbb{N} \mid x^e = y\}.$$

Definition 4 (batch proof of exponentiation)**.** A *batch proof of exponentiation* for m statements in a group $\mathbb{G}$ is an interactive proof for the language

$$L = \{\{(x_i, y_i, e)\}_{i\in[m]} \in \{\mathbb{G}^2 \times \mathbb{N}\}^m \mid x_i^e = y_i \text{ for all } i \in [m]\}.$$

In the paper, we sometimes use the term *batching protocol* to refer to a batch proof of exponentiation.

Known Batch Proofs of Exponentiation. Next, we present formally the Random Subsets Protocol by [7, 36] (Figure 2) and the Random Exponents Protocol by [36] (Figure 3) discussed on Section 1. Both of these protocols are adaptations of protocols from [6] to the setting of hidden order groups.

Optimizations of communication complexity. For ease of exposition, we present batching protocols in their interactive form. A non-interactive batch PoE can be obtained via the Fiat-Shamir heuristic [19], i.e., by deriving the verifier's challenges from the statements and current transcript via a suitable hash function. This is commonly done in constructions of PoE-based VDFs.

Parameters:
(1) group $\mathbb{G}$
(2) common exponent e
(3) number of statements m
(4) number of repetitions of subset multiplications ρ

Statements: $\left\{y_i \stackrel{?}{=} x_i^e\right\}_{i\in[m]}$ in $\mathbb{G}$

Protocol:
(1) $\mathcal{V}$ samples a matrix $B \leftarrow \{0,1\}^{\rho\times m}$ uniformly at random and sends it to $\mathcal{P}$.
(2) $\mathcal{V}$ and $\mathcal{P}$ both construct new statements $\left\{y'_i \stackrel{?}{=} (x'_i)^e\right\}_{i\in[\rho]}$, where
$$y'_i = \prod_{j\in[m]} y_j^{B_{i,j}} \text{ and } x'_i = \prod_{j\in[m]} x_j^{B_{i,j}}.$$
(3) $\mathcal{V}$ and $\mathcal{P}$ run ρ many PoE on statements $\left\{y'_i \stackrel{?}{=} (x'_i)^e\right\}_{i\in[\rho]}$ in parallel.

Figure 2: The Random Subsets Protocol [7, 36].

Parameters:
(1) group $\mathbb{G}$
(2) common exponent e
(3) number of statements m
(4) size of random coins ℓ

Statements: $\left\{y_i \stackrel{?}{=} x_i^e\right\}_{i\in[m]}$ in $\mathbb{G}$

Protocol:
(1) $\mathcal{V}$ samples a vector $r \leftarrow [2^\ell]^m$ uniformly at random and sends it to $\mathcal{P}$.
(2) $\mathcal{V}$ and $\mathcal{P}$ both construct one new statement $\tilde{y} \stackrel{?}{=} (\tilde{x})^e$, where
$$\tilde{y} = \prod_{i\in[m]} (y_i)^{r_i} \text{ and } \tilde{x} = \prod_{i\in[m]} (x_i)^{r_i}.$$
(3) $\mathcal{V}$ and $\mathcal{P}$ run PoE on statement $\tilde{y} \stackrel{?}{=} (\tilde{x})^e$.

Figure 3: The Random Exponents Protocol [36].

When interaction per se is not an issue, a natural approach for minimizing the communication complexity from the verifier to the prover suggested, e.g., in Rotem [36] is for the verifier to sample a key for a pseudorandom function (PRF), send it to the prover instead of the challenges, and let the prover derive the verifier's messages using the PRF. Similarly to the Fiat-Shamir heuristic, this optimization can be applied to any public-coin batching protocol and, in particular, to protocols presented in this work.

3 THE HYBRID PROTOCOL

In this section, we present our first protocol that improves the number of multiplications over the Random Exponents Protocol (Figure 3). We exploit the fact that the Random Subsets Protocol (Figure 2) results in λ instances that can be securely batched under

Parameters:

(1) group $\mathbb{G}$
(2) common exponent e
(3) number of statements m
(4) number of repetitions of subset multiplications ρ
(5) size of random coins ℓ

Statements: $\left\{y_i \stackrel{?}{=} x_i^e\right\}_{i\in[m]}$ in $\mathbb{G}$

Protocol:

(1) $\mathcal{V}$ samples a matrix $B \leftarrow \{0,1\}^{\rho\times m}$ and a vector $r \leftarrow [2^\ell]^\rho$ uniformly at random and sends both to $\mathcal{P}$.
(2) $\mathcal{V}$ and $\mathcal{P}$ both construct new statements $\left\{y'_i \stackrel{?}{=} (x'_i)^e\right\}_{i\in[\rho]}$, where
$$y'_i = \prod_{j\in[m]} y_j^{B_{i,j}} \text{ and } x'_i = \prod_{j\in[m]} x_j^{B_{i,j}}.$$
(3) $\mathcal{V}$ and $\mathcal{P}$ both construct one new statement $\tilde{y} \stackrel{?}{=} (\tilde{x})^e$, where
$$\tilde{y} = \prod_{i\in[\rho]} (y'_i)^{r_i} \text{ and } \tilde{x} = \prod_{i\in[\rho]} (x'_i)^{r_i}.$$
(4) $\mathcal{V}$ and $\mathcal{P}$ run PoE on statement $\tilde{y} \stackrel{?}{=} (\tilde{x})^e$.

Figure 4: Our Hybrid batch proof of exponentiation.

computational hardness assumptions. The resulting Hybrid Protocol reduces the number of PoEs needed to one while applying the random exponents technique solely to λ instances, i.e., the complexity of the more costly technique is made independent of the number of batched instances.

The protocol is presented in Figure 4. Completeness follows by inspection of the protocol. In the rest of the section, we prove its soundness based on the low order assumption (Definition 1).

Theorem 1. *Let* PoE *be a proof of exponentiation with soundness error γ and let $\mathbb{G}$ be a group output by* GGen(λ)*. Assuming the low order assumption for* GGen *with soundness error μ, the hybrid batching protocol presented in Figure 4 has soundness error at most $\gamma + \mu + 2^{-\rho} + 2^{-\lambda} + 2^{-\ell}$.*

Before proving Theorem 1, we state two lemmas that we will need in the proof. For the proof of Lemma 1, see Block et al. [7].

Lemma 1 ([7, Fact 8.1]). *For any group $\mathbb{G}$ and any $x_1, \ldots, x_n \in \mathbb{G}$, where at least one of them is not the identity element, we have*
$$\Pr_{S\leftarrow 2^{[n]}}\left[\prod_{i\in S} x_i = 1\right] \le \frac{1}{2}.$$

The second lemma is similar to [36, Lemma 6.2] but the proof differs. Importantly, Rotem's reduction incurs a quadratic security loss, and our reduction is tight.

Lemma 2. *Let $\mathbb{G}$ be a group and $n, e, \ell \in \mathbb{N}$ arbitrary positive integers. Let $\mathcal{A}$ be an algorithm that runs in time t and, with probability ε, outputs group elements $x_1, \ldots, x_n, y_1, \ldots, y_n \in \mathbb{G}$ such that, for at least one $i \in [n]$, we have $x_i^e \neq y_i$ and*
$$\Pr_{r_1,\ldots,r_n\leftarrow[2^\ell]}\left[\prod_{i\in[n]}\left(x_i^{r_i}\right)^e = \prod_{i\in[n]} y_i^{r_i}\right] = \frac{1}{2^\ell} + \delta.$$
Then there exists an algorithm $\mathcal{B}$ that runs in time $\mathrm{poly}(t, \log(e), n, \log(\delta^{-1})\delta^{-1})$ *and, with probability ε, outputs an element of order at most δ^{-1} in $\mathbb{G}$ together with its order.*

Proof. We let $\mathcal{B}$ run $\mathcal{A}$ and find one pair (x_{i^*}, y_{i^*}) for which $x_{i^*}^e \neq y_{i^*}$ among the group elements output by $\mathcal{A}$. This takes time $\mathrm{poly}(t, \log(e), n)$. Set $z = x_{i^*}^e / y_{i^*}$. We claim that z has order at most δ^{-1}. This means that $\mathcal{B}$ can find the order in time $\mathrm{poly}(\log(\delta^{-1})\delta^{-1})$ and output z together with its order. Clearly, $\mathcal{A}$ and $\mathcal{B}$ have the same success probability. It remains to prove the bound on the order of z. Let $(x_{i_1}, y_{i_1}), \ldots, (x_{i_s}, y_{i_s})$ be the pairs for which $x_{i_j}^e \neq y_{i_j}$. Assume without loss of generality that $x_{i_j}^e = y_{i_j} z_j$ for some group elements $z_1, \ldots, z_s \in \mathbb{G}$. Note that one of those elements is equal to z and denote the corresponding index by j^*. We have

$$\frac{1}{2^\ell} + \delta = \Pr_{r_1,\ldots,r_n\leftarrow[2^\ell]}\left[\prod_{i\in[n]}\left(x_i^{r_i}\right)^e = \prod_{i\in[n]} y_i^{r_i}\right] \tag{1}$$
$$= \Pr_{r_1,\ldots,r_n\leftarrow[2^\ell]}\left[\prod_{j\in[s]}\left(x_{i_j}^{r_{i_j}}\right)^e = \prod_{j\in[s]} y_{i_j}^{r_{i_j}}\right] \tag{2}$$
$$= \Pr_{r_1,\ldots,r_n\leftarrow[2^\ell]}\left[\prod_{j\in[s]}(y_{i_j} z_j)^{r_{i_j}} = \prod_{j\in[s]} y_{i_j}^{r_{i_j}}\right] \tag{3}$$
$$= \Pr_{r_1,\ldots,r_n\leftarrow[2^\ell]}\left[\prod_{j\in[s]} z_j^{r_{i_j}} = 1\right] \tag{4}$$
$$= \Pr_{r_1,\ldots,r_n\leftarrow[2^\ell]}\left[z^{r_{i^*}} = \prod_{j\in[s],j\neq j^*} z_j^{-r_{i_j}}\right]. \tag{5}$$

Equation (1) holds by assumption. In Equation (2), we only take the product over the incorrect statements, since the correct statements can be cancelled out in the equation. Equation (3) holds by definition of z_j's. Equation (4) follows from cancelling out correct statements. In Equation (5), we move all elements z_j different from z to the other side of the equation. The element on the right side of the equation is either an element of the subgroup generated by z or not. If it is not an element of the subgroup, the probability of the event is 0. Otherwise, let $\alpha < \mathrm{ord}(z)$ be such that
$$\prod_{j\in[s],j\neq j^*} z_j^{-r_{i_j}} = z^\alpha$$

and let s be the largest integer such that $s \cdot \mathrm{ord}(z) \leq 2^\ell$. Denote $\Pr := \Pr_{r_{i^*} \leftarrow [2^\ell]}$. Then we have

$$\begin{aligned}(5) &\leq \Pr\left[z^{r_{i^*}} = z^\alpha\right] \\ &= \Pr\left[z^{r_{i^*}} = z^\alpha \mid r_{i^*} \leq s \cdot \mathrm{ord}(z)\right] \cdot \Pr\left[r_{i^*} \leq s \cdot \mathrm{ord}(z)\right] \\ &\quad + \Pr\left[z^{r_{i^*}} = z^\alpha \mid r_{i^*} > s \cdot \mathrm{ord}(z)\right] \cdot \Pr\left[r_{i^*} > s \cdot \mathrm{ord}(z)\right] \\ &\leq \frac{s \cdot \mathrm{ord}(z)}{2^\ell} \cdot \frac{1}{\mathrm{ord}(z)} + \frac{2^\ell - s \cdot \mathrm{ord}(z)}{2^\ell} \cdot \frac{1}{2^\ell - s \cdot \mathrm{ord}(z)} \\ &\leq \frac{1}{\mathrm{ord}(z)} + \frac{1}{2^\ell}.\end{aligned}$$

Hence, we obtain that $2^{-\ell} + \delta \leq 1/\mathrm{ord}(z) + 2^{-\ell}$ and the claim follows. □

Proof of Theorem 1. Assume that at least one of the initial statements $\left\{y_i \stackrel{?}{=} x_i^e\right\}_{i\in[m]}$ is incorrect. An adversary that tries to break soundness of the protocol needs to be successful in one of the Steps 2,3,4, which means that either in Step 2 or 3 it starts with at least one wrong statement and at the end of the step all of the statements are correct or it breaks the soundness of the PoE in Step 4.

- By Lemma 1, we have that after Step 2 of the protocol at least one of the statements $\left\{y'_i \stackrel{?}{=} (x'_i)^e\right\}_{i\in[\rho]}$ is incorrect except with probability at most $1/2^\rho$.
- Assume that at least one of those statements is incorrect and consider Step 3. Applying Lemma 2 to the statements $\left\{y'_i \stackrel{?}{=} (x'_i)^e\right\}_{i\in[\rho]}$, we get that $\tilde{y} \stackrel{?}{=} (\tilde{x})^e$ is incorrect except with probability at most $\mu + 2^{-\lambda} + 2^{-\ell}$. Otherwise, one could find an element of order smaller than 2^λ with probability larger than μ, which contradicts the low order assumption.
- In Step 4 of the protocol $\mathcal{V}$ and $\mathcal{P}$ run a PoE with soundness error γ.

Taking the union bound, we get that the soundness error of the protocol is at most $\gamma + \mu + 2^{-\rho} + 2^{-\lambda} + 2^{-\ell}$ assuming the μ-low order assumption. □

On the quality of the low order assumption. As we have just seen in the proof of Theorem 1, the summand $2^{-\lambda}$ in our upper bound in the soundness error of the Hybrid Protocol comes from the assumption that it is hard to find elements of order less than 2^λ (see Definition 1). Note that a weaker variant of the low order assumption would suffice to get meaningful security. To weaken the assumption, one can restrict the bound on the order of elements the adversary can output while winning the game, for example, to $\lambda^{\log\lambda}$. This would increase the corresponding term in the bound on the soundness error from $2^{-\lambda}$ to $\lambda^{-\log\lambda}$, which is, however, still negligible. Similarly, a weaker variant of the low order assumption suffices also for our soundness analysis of the Bucket Protocol presented next. For simplicity of presentation, we stick to the bound of $2^{-\lambda}$ in the low order assumption throughout the rest of the paper.

Parameters:

(1) group $\mathbb{G}$
(2) common exponent e
(3) number of statements m
(4) number of buckets $K = 2^k$, where $k \leq \lambda$
(5) number of repetitions of bucketings $\rho = \lceil \lambda/(k-2) \rceil$
(6) size of random coins ℓ

Statements: $\left\{y_i \stackrel{?}{=} x_i^e\right\}_{i\in[m]}$ in $\mathbb{G}$

Protocol:

(1) $\mathcal{V}$ samples two matrices $B \leftarrow [K]^{\rho\times m}$ and $R \leftarrow [K]^{\rho\times K}$ and a vector $r \leftarrow [2^\ell]^\rho$ uniformly at random and sends both to $\mathcal{P}$.
(2) $\mathcal{V}$ and $\mathcal{P}$ both construct new statements $\left\{y'_{i,b} \stackrel{?}{=} (x'_{i,b})^e\right\}_{i\in[\rho],b\in[K]}$, where
$$y'_{i,b} = \prod_{j\in[m],B_{i,j}=b} y_j \text{ and } x'_{i,b} = \prod_{j\in[m],B_{i,j}=b} x_j.$$
(3) $\mathcal{V}$ and $\mathcal{P}$ both construct new statements $\left\{y''_i \stackrel{?}{=} (x''_i)^e\right\}_{i\in[\rho]}$, where
$$y''_i = \prod_{b\in[K]} (y'_{i,b})^{R_{i,b}} \text{ and } x''_i = \prod_{b\in[K]} (x'_{i,b})^{R_{i,b}}.$$
(4) $\mathcal{V}$ and $\mathcal{P}$ both construct one new statement $\tilde{y} \stackrel{?}{=} (\tilde{x})^e$, where
$$\tilde{y} = \prod_{i\in[\rho]} (y''_i)^{r_i} \text{ and } \tilde{x} = \prod_{i\in[\rho]} (x''_i)^{r_i}.$$
(5) $\mathcal{V}$ and $\mathcal{P}$ run PoE on statement $\tilde{y} \stackrel{?}{=} (\tilde{x})^e$.

Figure 5: Our Bucket batch proof of exponentiation based on the bucket test from Bellare et al. [6].

4 THE BUCKET PROTOCOL

In this section, we present an adaptation of the bucket test of Bellare, Garay, and Rabin [6] to the setting of batch proofs of exponentiation. The protocol is presented in Figure 5.

Starting with m statements, the following process is repeated $\rho = \lceil \lambda/(k-2) \rceil$ times: The prover and verifier uniformly at random place the m statements into $K = 2^k$ buckets and then compute the product of the statements in each bucket to obtain K new statements, which are aggregated to a single statement using the Random Exponents Protocol with k-bit exponents.

After the ρ repetitions, the prover and verifier aggregate the resulting ρ statements using the Random Exponents Protocol with λ-bit exponents, and they run a proof of exponentiation on the final statement. This last aggregation step differs from the bucket test in [6], where the ρ instances are verified directly in parallel instead of aggregating them to a single statement.

Note that both the prover and verifier must run in polynomial time in λ, and, thus, they could not keep track of 2^λ buckets. Therefore, without loss of generality, we use 2^λ as an upper bound on the number of buckets $K = 2^k$ in the proof of the following theorem.

Theorem 2. *Let* PoE *be a proof of exponentiation with soundness error γ and let $\mathbb{G}$ be a group output by* GGen*(λ). Assuming the low order assumption for* GGen *with soundness error $\mu \leq 2^{-k} - 2^{-\lambda}$, the bucket batching protocol presented in Figure 5 has soundness error at most $\gamma + \mu + 2^{-\lambda+1} + 2^{-\ell}$.*

Proof. Suppose that there is at least one incorrect statement among $\left\{y_i \stackrel{?}{=} x_i^e\right\}_{i\in[m]}$. An adversary that tries to break the soundness of the protocol needs to be successful in one of the Steps 2,3,4,5, which means that either in Step 2, 3 or 4 it starts with at least one wrong statement and at the end of the step all of the statements are correct or it breaks the soundness of the PoE in Step 5.

- Following the analysis in [6], we show that after Steps 2 and 3 at least one of the statements $\left\{y'_{i,b} \stackrel{?}{=} (x'_{i,b})^e\right\}_{i\in[\rho],b\in[K]}$ is incorrect except with probability $2^{-\lambda}$. Fix one round $i \in [\rho]$ and assume that all statements have been assigned to a bucket except for one incorrect statement. We call a bucket *good* if the product of all statements in that bucket yields a correct statement. Otherwise, we call it *bad*. The event that all K buckets are good after that round i can only occur if all but one bucket so far are good: If more than one bucket is bad, at least one of them cannot become good after assigning the final missing statement. If all buckets are good, assigning the final missing statement will make its bucket bad. In order for all buckets to be good in the end, we at least need that the final missing statement falls into the only bad bucket. This occurs with probability $1/K = 2^{-k}$.
- Now consider Step 3. Applying Lemma 2 to statements $\left\{y'_{i,b} \stackrel{?}{=} (x'_{i,b})^e\right\}_{b\in[K]}$ for a fixed $i \in [\rho]$, where at least one of the statements is incorrect, we get that the resulting statement $y''_i \stackrel{?}{=} (x''_i)^e$ is incorrect except with probability at most $\mu + 2^{-\lambda} + 2^{-k}$. Otherwise, one could find an element of order smaller than 2^λ with probability larger than μ, which contradicts the low order assumption. Taking the union bound, we get that at least one of the statements $\left\{y''_i \stackrel{?}{=} (x''_i)^e\right\}_{i\in[\rho]}$ is wrong except with probability at most $(\mu + 2^{-\lambda} + 2^{-k} + 2^{-k})^\rho \leq (2^{-k+2})^\rho \leq 2^{-\lambda}$ since $\mu \leq 2^{-k} - 2^{-\lambda}$ by assumption.
- Assume that at least one of those statements is incorrect and consider Step 4. Applying Lemma 2 to the statements $\left\{y''_i \stackrel{?}{=} (x''_i)^e\right\}_{i\in[\rho]}$, we get that $\tilde{y} \stackrel{?}{=} (\tilde{x})^e$ is incorrect, except with probability at most $\mu + 2^{-\lambda} + 2^{-\ell}$. Otherwise, one could find an element of order smaller than 2^λ with probability larger than μ, which contradicts the low order assumption.
- In Step 5 of the protocol $\mathcal{V}$ and $\mathcal{P}$ run a PoE with soundness error γ.

Taking the union bound, we get that the soundness error of the protocol is at most $\gamma + \mu + 2^{-\lambda+1} + 2^{-\ell}$. □

5 COMPARISON

A comparison of no batching and the four batching approaches is summarised in Table 1. The number of proofs to verify is clear in all approaches. Next, we explain the expected number of multiplications for each approach. This computation excludes the multiplication cost of the verification of the PoE proof(s) as that depends on the specific PoE.

Random Subsets: In expectation, the Random Subsets Protocol selects subsets of $[m]$ of size $m/2$, and, thus, it computes two products of size $m/2$ per each of the λ parallel repetitions.

Random Exponents: The m instances are raised to random λ-bit exponents, which is performed using $1.5\lambda \cdot 2m$ multiplications in expectation. Finally, it needs to compute two products of m group elements (to construct the resulting x'_i's and y'_i's).

Hybrid Protocol: In expectation, the protocol in Figure 4 performs the λm multiplications as in the Random Subsets Protocol. Then, it applies the Random Exponents Protocol to the resulting λ instances.

Bucket Protocol: The protocol in Figure 5 performs $\rho = \lceil \lambda/(k-2) \rceil$ repetitions. In total, it takes $2\rho m$ multiplications to produce the instances corresponding to the buckets as, in each repetition, every instance participates in exactly one bucket. Then, in each repetition, the $K = 2^k$ bucket instances are aggregated using the Random Exponents Protocol with coins of size k. Finally, the resulting ρ instances are merged using the Ranom Exponents Protocol with coins of size λ.

Note that the approaches are ordered on the basis of their efficiency. The Random Exponents Protocol increases the number of multiplications compared to the Random Subsets Protocol. However, it is the first protocol with a single PoE proof. The Hybrid Protocol achieves the same number of multiplications as the Random Subsets compiler up to an additive overhead independent of the number of instances m. Finally, the Bucket Protocol is parameterized by the number of buckets and enables a trade-off between the number of multiplications that depend on the number of instances and that are independent of the number of instances.

For security parameter $\lambda = 128$ and a varying number of PoE instances m, we plot the relative and total number of group multiplications in Figures 6 and 7. For the Bucket Protocol, we compute with the optimal parameter k w.r.t. λ and each m. Already at one thousand PoE instances, both the Hybrid Protocol and the Bucket Protocol significantly decrease the number of expected group multiplications compared to the Random Exponents Protocol. The Hybrid Protocol already achieves roughly a threefold decrease for tens of thousands of instances, which is the expected gain as can be seen from Table 1. Due to the varying k, the gap between the Random Exponents Protocol and our Bucket Protocol is increasing. It achieves a threefold improvement over the Random Exponents Protocol at a thousand instances, and it decreases the expected number of multiplications by an order of magnitude at a hundred thousand instances.

Table 1: The complexity of various batch PoEs for m instances with security parameter λ, and 2^k buckets in the Bucket Protocol.

Protocol	# multiplications	# proofs
No batching	0	m
Random Subsets	λm	λ
Random Exponents	$(3\lambda+2)m$	1
Hybrid	$\lambda(m+3\lambda+2)$	1
Bucket	$\left\lceil \frac{\lambda}{k-2} \right\rceil \left(2m+(3k+2)2^k+(3\lambda+2)\right)$	1

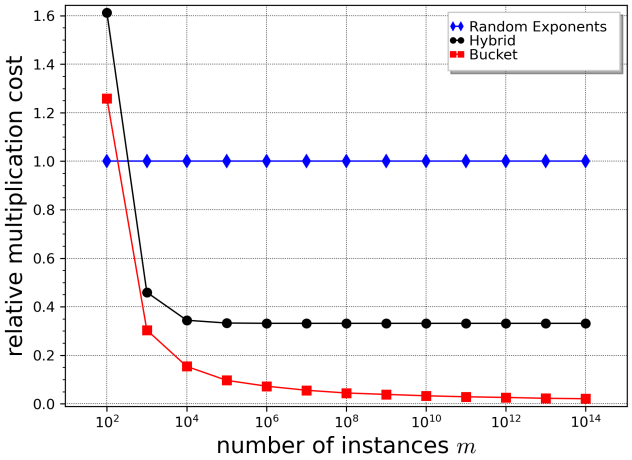


Figure 6: The relative number of multiplications on 100 to 10^{14} instances compared to the random exponent batching approach from Rotem [36] when the security parameter is set to $\lambda = 128$ and the parameter k in the Bucket Protocol is chosen optimally w.r.t. the number of instances.

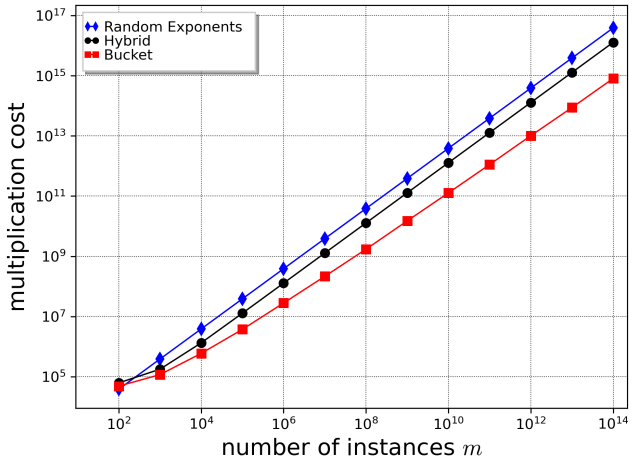


Figure 7: The absolute number of multiplications on 100 to 10^{14} instances when the security parameter is set to $\lambda = 128$ and the parameter k in the Bucket Protocol is chosen optimally w.r.t. the number of instances.

Table 2: Experimental evaluation of the times (in seconds) for PoE batching approaches in a 2048-bit RSA group. The last row is extrapolated for the first two approaches due to excessive times.

# instances	Random Exponents	Hybrid	Bucket
100	0.114	0.207	0.130
1 000	1.14	0.739	0.368
10 000	11.4	6.01	2.05
100 000	114	58.9	14.1
1 000 000	1140	584	109
10 000 000	*11400*	*5840*	892

5.1 Experimental Evaluation

In this section, we present experimental results that compare the practical performance of known approaches for PoE batching.

We have implemented all approaches in SageMath 10.1. Our implementation is available at [22]. We have timed the performance of the implementations on a machine with a four core 3.60GHz Intel Xeon CPU E5-1620 0 processor with 64 GB of RAM. For our experiments, we have generated 10M random PoE instances of the form $y = x^{2^{25}}$ with a common 2048-bit RSA modulus (roughly 12 GB of data). We then timed the performance of the Random Exponents Protocol, Hybrid Protocol, and Bucket protocol using the SageMath timeit function. The timing results on multiples of ten from one hundred to ten million instances are presented in Table 2 and Figures 8 and 9.

Discussion of experimental results. When comparing Figures 6 and 8, we see that the speed-up of the Bucket Protocol behaves as expected from the theoretical analysis: already at thousand instances, we see a significant speed-up, which, with a growing number of instances m quickly converges to a speed-up of one order of magnitude. However, the speed-up of the Hybrid Protocol is smaller than one could expect based solely on the number of group multiplications. It is roughly twice as fast as the Random Exponents protocol, whereas the number of multiplications is almost reduced by a factor of three. We conjecture that this is due to the complexity of creating the subsets in the first step of the protocol. In our implementation, we loop through λ many uniformly random bit strings of length m and always check if the current bit b is 0 or 1. We leave optimizing the implementation open for future work.

Leveraging efficient multi-exponentiation. In our experiments, we did not employ any advanced multi-exponentiation algorithm

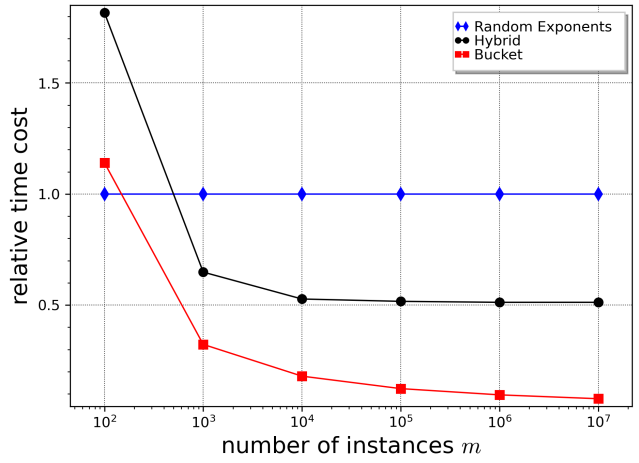


Figure 8: The relative time performance on 100 to 10M instances compared to the random exponent batching approach from Rotem [36], where the security parameter is set to $\lambda = 128$ and the parameter k is chosen optimally w.r.t. the number of instances.

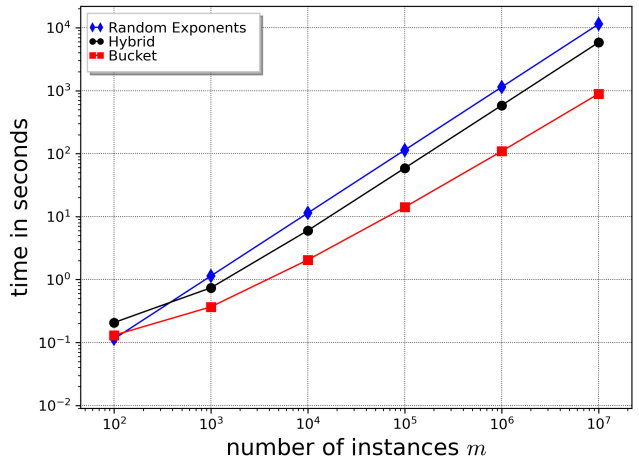


Figure 9: The times on 100 to 10M instances, where the security parameter is set to $\lambda = 128$ and the parameter k is chosen optimally w.r.t. the number of instances.

discussed in Section 1.2. We note that our Hybrid Protocol cannot take significant advantage of advanced multi-exponentiation algorithms. The bulk of its computation is to evaluate products of random subsets of the m instances, and it applies the Random Exponents Protocol always only to λ instances (i.e., roughly 128 instances). Therefore, it would achieve smaller advantage over an implementation of the Random Exponents Protocol that uses the best known multi-exponentiation algorithm for the specific number of instances and architecture.

However, the relative comparison of the Random Exponents Protocol and our Bucket Protocol should remain unaffected as any improvement in the complexity of multi-exponentiation that would speed up the former should provide similar gains also to the latter. Specifically, the parameter k in the Bucket protocol allows for a trade-off between the number of parallel repetitions $\rho = \lceil \lambda/(k-2) \rceil$ and the size of multi-exponentiations performed on the $K = 2^k$ buckets. Thus, faster multi-exponentiation would allow to increase the number of buckets and, correspondingly, decrease further the number of parallel repetitions.

6 CONCLUSIONS

In this paper, we have introduced two approaches to batch Proofs of Exponentiation – the Hybrid Protocol and the Bucket Protocol – both of which significantly improve over the state of the art.

The Hybrid Protocol is an easy to implement combination of the previously known Random Subsets and Random Exponents approaches. Compared to the Random Exponents Protocol, it yields a more efficient verification process when batching thousands or more instances. Through experimental validation, the Hybrid Protocol has shown to be approximately twice as fast as the Random Exponents Protocol.

The Bucket Protocol, drawing inspiration from the work of Bellare, Garay, and Rabin [6], further optimizes the batching process. This protocol, particularly effective for a large number of instances, demonstrates a significant reduction in computational overhead, outperforming the Random Exponents Protocol by an order of magnitude and the Random Subsets Protocol by a factor of five.

In conclusion, our work presents a meaningful advancement in the efficiency and practicality of batch PoEs, paving the way for more robust and efficient cryptographic protocols in distributed computing environments.

Various open problems are left for further research, such as:

- Aiming at even more efficient protocols, the pressing question is whether it is possible to construct batch PoEs with strictly linear or even *sublinear* verification.
- As we noted in Section 1.2, the known general batch PoEs are restricted to PoE statements with a shared exponent, while the batch PoE in [23] for instances with varying exponents inherently relies on Pietrzak's PoE. Thus, the construction of a generic batch PoE for instances with varying exponents remains open.

ACKNOWLEDGMENTS

Pavel Hubáček was supported by the Institute of Mathematics, Czech Academy of Sciences (RVO 67985840).

REFERENCES

[1] Arasu Arun, Joseph Bonneau, and Jeremy Clark. 2022. Short-lived Zero-Knowledge Proofs and Signatures. In *ASIACRYPT 2022, Part III (LNCS, Vol. 13793)*, Shweta Agrawal and Dongdai Lin (Eds.). Springer, Heidelberg, 487–516. https://doi.org/10.1007/978-3-031-22969-5_17

[2] Vidal Attias, Luigi Vigneri, and Vassil S. Dimitrov. 2020. Implementation Study of Two Verifiable Delay Functions. In *2nd International Conference on Blockchain Economics, Security and Protocols, Tokenomics 2020, October 26-27, 2020, Toulouse, France (OASIcs, Vol. 82)*, Emmanuelle Anceaume, Christophe Bisière, Matthieu Bouvard, Quentin Bramas, and Catherine Casamatta (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 9:1–9:14. https://doi.org/10.4230/OASICS.TOKENOMICS.2020.9

[3] Vidal Attias, Luigi Vigneri, and Vassil S. Dimitrov. 2022. Efficient Verification of the Wesolowski Verifiable Delay Function for Distributed Environments. *IACR Cryptol. ePrint Arch.* (2022), 520. https://eprint.iacr.org/2022/520

[4] Vidal Attias, Luigi Vigneri, and Vassil S. Dimitrov. 2023. Rethinking modular multi-exponentiation in real-world applications. *J. Cryptogr. Eng.* 13, 1 (2023), 57–70. https://doi.org/10.1007/S13389-022-00287-W

[5] Karim Belabas, Thorsten Kleinjung, Antonio Sanso, and Benjamin Wesolowski. 2020. A note on the low order assumption in class group of an imaginary quadratic number fields. *IACR Cryptol. ePrint Arch.* (2020), 1310. https://eprint.iacr.org/2020/1310

[6] Mihir Bellare, Juan A. Garay, and Tal Rabin. 1998. Fast Batch Verification for Modular Exponentiation and Digital Signatures. In *EUROCRYPT'98 (LNCS, Vol. 1403)*, Kaisa Nyberg (Ed.). Springer, Heidelberg, 236–250. https://doi.org/10.1007/BFb0054130

[7] Alexander R. Block, Justin Holmgren, Alon Rosen, Ron D. Rothblum, and Pratik Soni. 2021. Time- and Space-Efficient Arguments from Groups of Unknown Order. In *CRYPTO 2021, Part IV (LNCS, Vol. 12828)*, Tal Malkin and Chris Peikert (Eds.). Springer, Heidelberg, Virtual Event, 123–152. https://doi.org/10.1007/978-3-030-84259-8_5

[8] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. 2018. Verifiable Delay Functions. In *CRYPTO 2018, Part I (LNCS, Vol. 10991)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Heidelberg, 757–788. https://doi.org/10.1007/978-3-319-96884-1_25

[9] Dan Boneh, Benedikt Bünz, and Ben Fisch. 2018. A Survey of Two Verifiable Delay Functions. Cryptology ePrint Archive, Report 2018/712. https://eprint.iacr.org/2018/712.

[10] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. 2012. Batch Verification of Short Signatures. *J. Cryptol.* 25, 4 (2012), 723–747. https://doi.org/10.1007/S00145-011-9108-Z

[11] Jorge Chavez-Saab, Francisco Rodríguez-Henríquez, and Mehdi Tibouchi. 2022. Verifiable Isogeny Walks: Towards an Isogeny-Based Postquantum VDF. In *Selected Areas in Cryptography*, Riham AlTawy and Andreas Hülsing (Eds.). Springer International Publishing, Cham, 441–460.

[12] Valerio Cini, Russell W. F. Lai, and Giulio Malavolta. 2023. Lattice-Based Succinct Arguments from Vanishing Polynomials - (Extended Abstract). In *CRYPTO 2023, Part II (LNCS, Vol. 14082)*, Helena Handschuh and Anna Lysyanskaya (Eds.). Springer, Heidelberg, 72–105. https://doi.org/10.1007/978-3-031-38545-2_3

[13] Jeremy Clark and Aleksander Essex. 2012. CommitCoin: Carbon Dating Commitments with Bitcoin - (Short Paper). In *FC 2012 (LNCS, Vol. 7397)*, Angelos D. Keromytis (Ed.). Springer, Heidelberg, 390–398.

[14] Bram Cohen and Krzysztof Pietrzak. 2018. Simple Proofs of Sequential Work. In *EUROCRYPT 2018, Part II (LNCS, Vol. 10821)*, Jesper Buus Nielsen and Vincent Rijmen (Eds.). Springer, Heidelberg, 451–467. https://doi.org/10.1007/978-3-319-78375-8_15

[15] Bram Cohen and Krzysztof Pietrzak. 2019. *The Chia Network Blockchain.* Technical Report. Chia Network. https://chia-network.github.io/assets/ChiaGreenPaper.pdf

[16] Giovanni Di Crescenzo, Matluba Khodjaeva, Delaram Kahrobaei, and Vladimir Shpilrain. 2017. Computing multiple exponentiations in discrete log and RSA groups: From batch verification to batch delegation. In *2017 IEEE Conference on Communications and Network Security, CNS 2017, Las Vegas, NV, USA, October 9-11, 2017*. IEEE, 531–539. https://doi.org/10.1109/CNS.2017.8228702

[17] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. 2019. Verifiable Delay Functions from Supersingular Isogenies and Pairings. In *ASIACRYPT 2019, Part I (LNCS, Vol. 11921)*, Steven D. Galbraith and Shiho Moriai (Eds.). Springer, Heidelberg, 248–277. https://doi.org/10.1007/978-3-030-34578-5_10

[18] Amos Fiat. 1997. Batch RSA. *J. Cryptol.* 10, 2 (1997), 75–88. https://doi.org/10.1007/S001459900021

[19] Amos Fiat and Adi Shamir. 1987. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO'86 (LNCS, Vol. 263)*, Andrew M. Odlyzko (Ed.). Springer, Heidelberg, 186–194. https://doi.org/10.1007/3-540-47721-7_12

[20] Cody Freitag, Rafael Pass, and Naomi Sirkin. 2022. Parallelizable Delegation from LWE. In *TCC 2022, Part II (LNCS, Vol. 13748)*, Eike Kiltz and Vinod Vaikuntanathan (Eds.). Springer, Heidelberg, 623–652. https://doi.org/10.1007/978-3-031-22365-5_22

[21] Ryan Henry. 2010. *Pippenger's Multiproduct and Multiexponentiation Algorithms.* Technical Report. University of Waterloo. https://cacr.uwaterloo.ca/techreports/2010/cacr2010-26.pdf

[22] Charlotte Hoffmann, Pavel Hubáček, and Svetlana Ivanova. 2024. Implementation of experiments. https://github.com/pavelhubacek/batchPoEs

[23] Charlotte Hoffmann, Pavel Hubáček, Chethan Kamath, Karen Klein, and Krzysztof Pietrzak. 2022. Practical Statistically-Sound Proofs of Exponentiation in Any Group. In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 13508)*, Yevgeniy Dodis and Thomas Shrimpton (Eds.). Springer, 370–399. https://doi.org/10.1007/978-3-031-15979-4_13

[24] Charlotte Hoffmann, Pavel Hubáček, Chethan Kamath, and Tomáš Krňák. 2023. (Verifiable) Delay Functions from Lucas Sequences. In *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part IV (Lecture Notes in Computer Science, Vol. 14372)*, Guy N. Rothblum and Hoeteck Wee (Eds.). Springer, 336–362. https://doi.org/10.1007/978-3-031-48624-1_13

[25] Charlotte Hoffmann, Pavel Hubáček, Chethan Kamath, and Krzysztof Pietrzak. 2023. Certifying Giant Nonprimes. In *Public-Key Cryptography - PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 13940)*, Alexandra Boldyreva and Vladimir Kolesnikov (Eds.). Springer, 530–553. https://doi.org/10.1007/978-3-031-31368-4_19

[26] Chia Network Inc. 2023. Chia Network. http://chia.net. Accessed: 2023-12-19.

[27] Dmitry Khovratovich, Mary Maller, and Pratyush Ranjan Tiwari. 2022. MinRoot: Candidate Sequential Function for Ethereum VDF. Cryptology ePrint Archive, Report 2022/1626. https://eprint.iacr.org/2022/1626.

[28] Russell W. F. Lai and Giulio Malavolta. 2023. Lattice-Based Timed Cryptography. In *CRYPTO 2023, Part V (LNCS, Vol. 14085)*, Helena Handschuh and Anna Lysyanskaya (Eds.). Springer, Heidelberg, 782–804. https://doi.org/10.1007/978-3-031-38554-4_25

[29] Esteban Landerreche, Marc Stevens, and Christian Schaffner. 2020. Non-interactive Cryptographic Timestamping Based on Verifiable Delay Functions. In *FC 2020 (LNCS, Vol. 12059)*, Joseph Bonneau and Nadia Heninger (Eds.). Springer, Heidelberg, 541–558. https://doi.org/10.1007/978-3-030-51280-4_29

[30] Gaëtan Leurent, Bart Mennink, Krzysztof Pietrzak, and Vincent Rijmen. 2023. *Analysis of MinRoot: Public report (requested by Ethereum Foundation).* Technical Report. Ethereum Foundation. https://crypto.ethereum.org/events/minrootanalysis2023.pdf

[31] Bodo Möller. 2001. Algorithms for Multi-exponentiation. In *Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers (Lecture Notes in Computer Science, Vol. 2259)*, Serge Vaudenay and Amr M. Youssef (Eds.). Springer, 165–180. https://doi.org/10.1007/3-540-45537-X_13

[32] Krzysztof Pietrzak. 2019. Simple Verifiable Delay Functions. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA (LIPIcs, Vol. 124)*, Avrim Blum (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 60:1–60:15. https://doi.org/10.4230/LIPIcs.ITCS.2019.60

[33] Nicholas Pippenger. 1980. On the Evaluation of Powers and Monomials. *SIAM J. Comput.* 9, 2 (1980), 230–250. https://doi.org/10.1137/0209022

[34] Michael O. Rabin. 1983. Transaction Protection by Beacons. *J. Comput. Syst. Sci.* 27, 2 (1983), 256–267.

[35] R. L. Rivest, A. Shamir, and D. A. Wagner. 1996. *Time-lock Puzzles and Timed-release Crypto.* Technical Report. Massachusetts Institute of Technology, Cambridge, MA, USA.

[36] Lior Rotem. 2021. Simple and Efficient Batch Verification Techniques for Verifiable Delay Functions. In *TCC 2021, Part III (LNCS, Vol. 13044)*, Kobbi Nissim and Brent Waters (Eds.). Springer, Heidelberg, 382–414. https://doi.org/10.1007/978-3-030-90456-2_13

[37] Philipp Schindler, Aljosha Judmayer, Markus Hittmeir, Nicholas Stifter, and Edgar R. Weippl. 2021. RandRunner: Distributed Randomness from Trapdoor VDFs with Strong Uniqueness. In *NDSS 2021*. The Internet Society.

[38] Kineret Segal and Tom Brand. 2019. *Presenting: VeeDo a STARK-based VDF Service.* Technical Report. StarkWare. https://medium.com/starkware/presenting-veedo-e4bbff77c7ae

[39] István András Seres and Péter Burcsi. 2020. A Note on Low Order Assumptions in RSA groups. *IACR Cryptol. ePrint Arch.* (2020), 402. https://eprint.iacr.org/2020/402

[40] Barak Shani. 2019. A note on isogeny-based hybrid verifiable delay functions. Cryptology ePrint Archive, Paper 2019/205. https://eprint.iacr.org/2019/205 https://eprint.iacr.org/2019/205.

[41] Ernst G. Straus. 1964. Addition chains of vectors (problem 5125). *Amer. Math. Monthly* 70, 806-808 (1964), 16.

[42] Joachim von zur Gathen and Jürgen Gerhard. 2013. *Modern Computer Algebra (3. ed.).* Cambridge University Press.

[43] Benjamin Wesolowski. 2019. Efficient Verifiable Delay Functions. In *EUROCRYPT 2019, Part III (LNCS, Vol. 11478)*, Yuval Ishai and Vincent Rijmen (Eds.). Springer, Heidelberg, 379–407. https://doi.org/10.1007/978-3-030-17659-4_13