The Cost of Statistical Security in Interactive Proofs for Repeated Squaring

Cody Freitag[*] Ilan Komargodski[†]

Abstract

In recent years, the number of applications of the repeated squaring assumption has been growing rapidly. The assumption states that, given a group element x, an integer T, and an RSA modulus N, it is hard to compute $x^{2^T} \bmod N$—or even decide whether $y \overset{?}{=} x^{2^T} \bmod N$—in parallel time less than the trivial approach of computing T sequential squarings. This rise has been driven by efficient interactive proofs for repeated squaring, opening the door to more efficient constructions of verifiable delay functions, various secure computation primitives, and proof systems for more general languages.

In this work, we study the complexity of *statistically-sound* interactive proofs for the repeated squaring relation. Technically, we consider interactive proofs where the prover sends at most $k \geq 0$ elements per round and the verifier performs generic group operations over the group $\mathbb{Z}_N^\star$. As our main contribution, we show that for any one-round proof with a randomized verifier (i.e., an MA proof) the verifier either runs in parallel time $\Omega(T/(k+1))$ with high probability, or is able to factor N given the proof provided by the prover. This shows that either the prover essentially sends p, q such that $N = p \cdot q$ (which is infeasible or undesirable in most applications), or a variant of Pietrzak's proof of repeated squaring (ITCS 2019) has optimal verifier complexity $O(T/(k+1))$. In particular, it is impossible to obtain a statistically-sound one-round proof of repeated squaring with efficiency on par with the computationally-sound protocol of Wesolowski (EUROCRYPT 2019), with a generic group verifier. We further extend our one-round lower bound to a natural class of recursive (multi-round) interactive proofs for repeated squaring.

[*]Cornell Tech and NTT Research, `cfreitag@cs.cornell.edu`
[†]Hebrew University and NTT Research, `ilank@cs.huji.ac.il`

Contents

1 Introduction

An interactive proof system is a method that allows an all powerful but untrusted prover to convince a resource-limited verifier of the validity of a computational statement. Since their introduction by Goldwasser, Micali, and Rackoff [GMR89] and Babai [Bab85], our understanding of proof systems has significantly evolved and has led to groundbreaking results in several different areas of computer science such as the PCP theorem, hardness of approximation results, zero-knowledge proofs, and more.

Interactive proofs are extremely expressive and this is largely due to interaction and randomness. Indeed, the class of languages that have an interactive proof is characterized by PSPACE [LFKN92, Sha90]. On the other hand, if we limit the prover to send only one message, this yields the class AM that seems much closer to NP. The question of how efficient proof systems could be has drawn significant attention over the years, both due to theoretical motivation of understanding the limits of such systems as well as due to practical needs as some of these schemes are implemented and deployed.

In this work, we study the efficiency of interactive proof systems for a particular language that has received significant interest in recent years due to its many exciting applications. This language is the repeated squaring language, defined with respect to a multiplicative group of integers modulo N:

$$\mathcal{RS}_N = \left\{ (x, y, T) \mid y = x^{2^T} \bmod N \right\},$$

where x and y are two group elements and T is an integer.

This problem is efficiently decidable by sequentially squaring T times given x and checking whether the result equals y. That is, $\mathcal{RS}_N \in \mathsf{DTime}(T \cdot \mathsf{poly} \log N)$. If $\varphi(N)$, the order of the group, is known, then $x^{2^T} \bmod N$ can be computed much faster by computing $x^{2^T \bmod \varphi(N)} \bmod N$. Rivest, Shamir, and Wagner [RSW96] conjectured that, unless one knows $\varphi(N)$, the sequential algorithm of computing T squarings is optimal and there is no better algorithm, even if significant parallel processing power is available. A compelling explanation for the lack of progress on this conjecture in the past 3 decades was given in two independent recent works by Katz, Loss, and Xu [KLX20] and Rotem and Segev [RS20]. Both results show (in different and incomparable models) that "generic" improvements to the trivial T-step sequential algorithm will directly yield a better algorithm for factoring N and hence computing $\varphi(N)$.

In their original work, Rivest et al. [RSW96] assumed the hardness of the repeated squaring language in hidden order groups in order to build a cryptographic analogue of a "time capsule" called a time-lock puzzle, namely, a mechanism for sending messages to the future. Since then, this assumption has been instrumental in the design of many cryptographic primitives, including various variants and extensions of time-lock puzzles [BN00, Pie19, Wes20, MT19, EFKP20a, DKP21, FKPS21] as well as other seemingly unrelated applications such as round-efficient non-malleable commitment [LPS20, BL18] and secure computation protocols [FJK21].

Interactive proofs for $\mathcal{RS}$. Several of the applications of the hardness of the $\mathcal{RS}$ language are enabled by leveraging (public-coin[1]) interactive proofs for $\mathcal{RS}$ in some non-trivial way. These applications include verifiable delay functions (VDFs) [BBBF18, Pie19, Wes20], fair multiparty

[1]In the public-coin model, at each round, the verifier tosses a predetermined number of coins and sends the outcome to the prover. We focus only on this model.

coin-flipping protocols [FKPS21], polynomial commitment schemes [BFS20], and highly efficient zero-knowledge arguments for NP [BHR$^+$21].

There are basically two classes of interactive proofs for repeated squaring:

- *Statistically sound proofs*: these proofs guarantee soundness for any, even unbounded, cheating prover. This is the stronger notion of soundness and there are applications that necessarily rely on it (see more on this below).

  Reingold, Rothblum, and Rothblum [RRR21] give a generic interactive proof system for polynomial-time and bounded space computations. This yields a constant-round interactive proof with constant statistical soundness. Specifically, for any $\delta \in (0, 1/2)$, there is a protocol with communication complexity and verifier's running time are $T^{\delta} \cdot \mathsf{poly} \log N$.

  Pietrzak [Pie19] gave a specialized and significantly more efficient statistically sound interactive proof for $\mathcal{RS}$ using its group structure. Viewing it as an r-round protocol, the prover in this protocol sends r group elements in total and in each round the verifier responds with a single, uniformly random, λ-bit integer. In the final round, the verifier runs in time $O(T/2^r)$ to verify the relation. If $r = \log T$, then the prover only communicated $\log T$ group elements. The prover only performs roughly T group operations and the verifier also performs very few group operations on the observed transcript.

  Exponentially small (in λ) statistical soundness holds only if the underlying group has no small order subgroups (e.g., Pietrzak [Pie19] considered the group QR_N^+ of signed quadratic residues modulo $N \in \mathbb{N}$, which is the product of two safe primes, and Rotem [Rot21] later extended the protocol to $\mathbb{Z}_N^\star$ for general $N = p \cdot q$ without small order subgroups). This assumption was lifted and a protocol for arbitrary hidden order groups was given by Block et al. [BHR$^+$21], albeit with an additional cost in efficiency.

- *Computationally sound proofs*: these are proofs that guarantee soundness only for computationally bounded attackers. While providing a weaker security guarantee, such proofs are traditionally (asymptotically and concretely) much more efficient.

  Such concrete schemes for $\mathcal{RS}$ were first given by Boneh et al. [BBBF18], relying on generic succinct computationally sound proof [Kil92, Mic94, Val08] (see also [EFKP20b]). Being generic, this approach still results with concretely inefficient protocols and is therefore mostly of theoretical interest. A much more efficient scheme, given by Wesolowski [Wes20], requires only a single round of interaction with proof size being a single group element and where the verifier only performs $O(\lambda)$ group operations, where λ is a security parameter, independent of T. However, it requires a strong (and novel) hardness assumption called the adaptive root assumption.

Overall, the existing computationally sound proofs are more efficient than their statistical counterparts in theory and in practice. However, in some applications statistical soundness is necessary. One reason could be that it is unacceptable to rely on strong or novel assumptions.[2] For instance,

[2]In applications, interactive proofs are often made non-interactive using the Fiat-Shamir transform [FS86]. This in turn adds another assumption to the system (either the random oracle model or that of a cryptographic hash function). The security of instantiating the Fiat-Shamir transform with computationally sound proofs is not very well understood; there are even examples of systems where it provably fails [Bar01, GK03]. In comparison, for statistically sound proofs we can often instantiate it using standard assumptions (e.g., [CCH$^+$19, LV20]).

in a blockchain setting where the stakes are high, it might not be ideal to design a critical component of the system with security relying on new and untested assumptions. Another reason is that there are situations where relying on a computationally sound proof for $\mathcal{RS}$ actually results with an insecure system.

Imagine that we compile a proof system into a VDF[3] in order to generate a randomness beacon for a blockchain (as suggested by Boneh et al. [BBBF18]). An important question is how the modulus N is chosen. In practice, a specialized distributed protocol is executed among few participants with stake.[4] If these participants later decide to become rouge, they can recover the order of the group at hand. Clearly, this allows them to compute exponents much faster than other participants and therefore "see into the future". However, if the underlying interactive proof was an argument system, then the situation would be much worse. Since for them the group is no longer of hidden order, they can potentially generate accepting proofs for false statements, thereby allowing them to "rewrite history", the fundamental attack that blockchains were designed to prevent. We emphasize that if the underlying interactive proof is statistically sound, then this problem does not exist as it is *impossible* to generate accepting proofs for false statements (this is true even if $P = NP$ and factoring is easy).

A very related scenario actually comes up in the fair multiparty coin-flipping protocol of Freitag et al. [FKPS21]. At a high level, Freitag et al. introduced a notion called *strong trapdoor VDFs*; these are VDFs that can be computed fast if a trapdoor is known. They needed such a primitive in order to prove the correctness of solving a time-lock puzzle (so that other parties in the system do not need to re-solve it over and over again). A delicate point is that the participant who generated the time-lock puzzle could be the one who also solves the VDF, meaning that they could know a trapdoor. If VDF proofs that are verified yet they correspond to a false statement exist, the security of their protocol is completely lost. Therefore, they had to rely on an underlying statistically sound interactive proof for $\mathcal{RS}$.

This raises the following very natural question which is the focus of our work:

What is the cost of statistically sound proof for $\mathcal{RS}$? Is Pietrzak's protocol [Pie19] optimal? Ideally, can we build such a scheme which is as efficient as the computationally sound scheme of Wesolowski [Wes20]?

Given that the uses of such interactive proofs have become more common and diverse, this urges us to explore and obtain a rigorous understanding of the limits of such systems.

1.1 Our Results

We make progress towards resolving the above-mentioned questions. Within a certain restricted model (the generic-group model relative to a hidden order group; see below), we prove two results on the tradeoffs between the communication complexity and the verifier's complexity in a large class of interactive proofs for $\mathcal{RS}$. Our results imply that for some class of protocols that we consider, any improvement over known protocols would lead to a non-trivial factoring algorithm.

[3]A VDF is a function that requires some "long" time T to compute (where T is a parameter given to the function), yet the answer to the computation can be efficiently verified given a proof that can be jointly generated with the output (with only small overhead). VDFs can be obtained by applying the Fiat-Shamir transform [FS86] on a given interactive proof for $\mathcal{RS}$ [Pie19, Wes20].

[4]For example, ZCash conducted a so called "ceremony" for the occasion with 6 publicly undisclosed participants (see `https://z.cash/technology/paramgen/`).

Thus, assuming that factoring is hard, any improvement must either be outside of the restricted model or relax soundness to computational.

It may seem odd that we could prove a lower bound on the complexity of proof systems for $\mathcal{RS}$ since the verifier could potentially decide $\mathcal{RS}$ without the prover's help at all. This can be done in one of the following ways. We rule out such (not interesting) proofs by limiting the power of the verifier, as follows.

- The verifier could compute $x^{2^T} \bmod N$ by first computing $\varphi(N)$, say by factoring N.

  We get around this by considering only *generic-group* verifiers. Recall that generic-group verifiers can perform any group operation while ignoring any specific property of the representation of elements. Rotem and Segev [RS20] show in this model that any non-trivial method for computing $x^{2^T} \bmod N$ directly implies a factoring algorithm in the plain model. (We remark that the prover in our proof systems need not be generic.)

- The verifier could directly compute $x^{2^T} \bmod N$ via T repeated squarings.

  We get around this by placing a restriction on the verifier's computational complexity. Specifically, we consider polynomial-time verifiers with bounded *parallel* running time. This allows us to capture parties that have hefty parallel processing power.

A bound for MA proofs. First, we consider 1-message protocols where the prover sends the verifier a single possibly long message, and then the verifier decides whether to accept or not by running a probabilistic polynomial time computation. This corresponds to the class MA (which generalizes NP by allowing the verifier to be probabilistic). There are basically two extreme interactive proofs one may consider:

1. The prover (somehow) learns $\varphi(N)$ and sends it to the verifier. The verifier can compute $x^{(2^T \bmod \varphi(N))} \bmod N$. In this case, V is deterministic and its running time is basically independent of T.

2. This protocol is parametrized by the communication complexity k and is an adaptation of Pietrzak's protocol [Pie19]. The prover computes $k \geq 0$ midpoints: $x_1 = x^{2^{T/(k+1)}}, x_2 = x^{2^{2T/(k+1)}}, \ldots, x_k = x^{2^{kT/(k+1)}}$ and sends $x_1, \ldots, x_k$ to the verifier. The verifier computes a random linear combination $z = \prod_{i=1}^{k+1}((x_{i-1})^{r_i})^{T/(k+1)}$ (with $x_0 = x$ and $x_{k+1} = y$) and finally accepts if $z = \prod_{i=1}^{k+1}(x_i)^{r_i}$. By a similar proof to that of Pietrzak [Pie19] (see also [EFKP20a]), with high probability over the r_i's, the equality holds if and only if all of the midpoints were computed correctly. Overall, the communication consists of k group elements and the verifier's running time is $O(T/(k+1))$.

We show that the above two protocols are essentially the best possible among all generic-group MA proofs. Specifically, we show that *either* we can factor composite numbers (in which case the first protocol can be constructed), or otherwise in any MA proof that includes $k \geq 0$ group elements, the verifier must run in parallel time at least $\Omega(T/(k+1))$. Additionally, if neither of these hold, then the protocol must not be statistically sound—there must exist proofs for false statements, even if they may be computationally hard to find.

We prove our result by presenting an algorithm that uses any too-good-to-be-true generic-group MA proof to solve factoring in the plain model. To this end, we use Maurer's [Mau05] generic-group algorithms abstraction and extend it to capture MA proofs. In our model, we restrict the verifier to be a generic-group algorithm (in Maurer's sense) that makes a bounded number of group multiplication and division queries, and we say that it accepts if it outputs the group's identity 1. Notice, for example, that this allows the verifier to compute two element g_1, g_2 and accept if they are equal by outputting $g_1 \cdot g_2^{-1}$. All efficient proofs specifically designed for $\mathcal{RS}$ fall into this model. The prover, on the other hand, may still be an unbounded (not necessarily generic) algorithm whose proof consists of a bit string and a sequence of group elements. Note that *not* restricting the prover to be generic only makes our result applicable to larger classes of constructions, thereby making it stronger. Refer to Section 3 for the precise model definition. We emphasize that even in this simplified one-round setting, it turns out to be highly non-trivial to prove our result in a way that captures the behaviors of arbitrary provers and verifiers; see Section 1.3 for an overview.

Theorem 1 (Simplified and Informal; see Theorem 3). *For any generic-group MA proof system for $\mathcal{RS}_N$, if the prover sends $k \geq 0$ group elements and a string* st*, the verifier either runs in parallel time $\Omega(T/(k+1))$, or is able to factor N given* st*.*

In fact, we prove in Corollary 1 that the above holds for any hidden order group. In the general case, we show that either the verifier runs in parallel time $\Omega(T/(k+1))$, or is able to compute (a multiple of) the order of the group given the string st output by the prover. However, by a variant of the Miller-Rabin primality test [Mil76, Rab80], it is well known that this immediately implies a factoring algorithm for N.

We note that if the prover is efficient, we can compute st ourselves. So, the existence of a verifier with $o(T/(k+1))$ parallel runtime implies a standard model factoring algorithm.

A bound for IPs. We extend our lower bound for MA proofs to a certain natural class of general (multi-round) interactive proofs. Specifically, we consider a class of *recursive IPs*, where in every round of communication, the prover attempts to prove a new instance of $\mathcal{RS}_N$, although with a different starting point x, a different endpoint y, and a different delay parameter T. Here, for a bound on the round complexity r and a communication bound k, the adaptation of Pietrzak's [Pie19] protocol results with a recursive IP with total communication $k \cdot r$ and verifier's running time $O(T/(k+1)^r)$. Here, we obtain a nearly optimal tradeoff between the message complexity, the round complexity, and the verifier's parallel running time.

Theorem 2 (Simplified and Informal; see Theorem 4). *For any generic-group r-round recursive interactive proof system for $\mathcal{RS}_N$, if the prover sends k group elements per round and results in a transcript* tr*, the verifier either runs in parallel time $\Omega(T/(2(k+1))^r)$, or is able to factor N given* tr*.*

Future Directions and Open Problems

Our work leaves many exciting open problems. We mention some of them next:

1. We prove our result in the generic-group model where we only allow multiplication and division queries. It would be interesting to extend this to handle general equality queries or addition/subtraction queries in the the generic-ring model [AM16, JS13, RS20].

2. Can we get a similar result to Theorem 2 for general (public-coin) IPs rather than just for "recursive" IPs?

3. Our bound from Theorem 2 says that (assuming factoring is hard) the verifier's parallel running time must be $\Omega(T/(2^r(k+1)^r))$ if the prover is efficient, while in Pietrzak's protocol [Pie19] the running time is $O(T/((k+1)^r))$ (and has an efficient prover). The extra 2^{-r} term in our bound comes from a factor of 2 loss in Theorem 1. It would be interesting to prove a tight bound (up to constants) for Theorem 1 for this reason.

1.2 Related Work

Complexity of interactive proofs. Goldreich and Håstad [GH98] initiated the investigation of interactive proofs with bounded communication. They showed that if a language L has an interactive proof in which the total communication is bounded by $c(n)$ bits then $L \in \mathsf{BPTime}(2^{c(n)} \cdot \mathsf{poly(n)})$. Further relations between the communication complexity of interactive proof for a language and its complement were shown by Goldreich, Vadhan, and Widgerson [GVW02].

The IP=PSPACE result [LFKN92, Sha90] says that languages that can be verified in polynomial time are exactly those proofs that can be generated with polynomial space. In this interactive proof system, the honest prover runs in super-polynomial time (even for log-space languages); this is true even for the scaled down version which captures polynomially recognizable languages. Nevertheless, the "easy" side of this result says that every language with an interactive proof of c bits is decidable with c space [LFKN92, Sha90]. Therefore, languages that require a lot of space to decide cannot have super efficient interactive proof systems.

Computationally sound proof systems can recognize any language in NP while using only polylogarithmic message complexity (assuming collision resistant hash functions) [Kil92].

In the statistical setting, the first interactive proofs with an *efficient* prover were given by Goldwasser, Kalai, and Rothblumn [GKR15]. They designed an interactive proof system where the honest prover is efficient and run in polynomial time. In their proof system the language is given by a log-space uniform Boolean circuit with depth d and input length n. Their verifier runs in time $n \cdot \mathsf{poly}(d, \log n)$, the communication complexity is $\mathsf{poly}(d, \log n)$, and the prover runs in time $\mathsf{poly}(n)$. This protocol is very useful for low-depth computations.

Reingold, Rothblum, and Rothblum [RRR21] showed a different protocol which suits polynomial time and bounded-polynomial space computations. They give a constant round protocol for polynomial time and space $S = S(n)$ languages such that: the honest prover runs in polynomial time, the verifier is almost linear time, and the communication complexity is $O(S \cdot n^\delta)$ for $\delta \in (0, 1)$. Applied on the repeated squaring language, (where $S = \mathsf{poly}\log n$) this protocol's communication roughly matches Pietrzak's [Pie19] when adapted to run in constant rounds (in which case it also requires the transmission of n^δ group elements).

Generic models. The problem we consider can be placed in a long line of research on proving efficiency trade-offs for various primitives, in some restricted class of constructions usually termed "black-box" or "generic". Generic or black-box constructions have the benefit of being applicable to every instantiation of the underlying structure, irrespectively of the exact details of its description. For specific instances, this usually allows for cleaner and more efficient constructions. The interactive proofs for $\mathcal{RS}$ of Pietrzak [Pie19] and Wesolowski [Wes20] are generic.

Our work is the first to study the complexity of interactive proofs for $\mathcal{RS}$ from a foundational perspective. The most relevant previous works study the ("generic") complexity of related cryptographic primitives or assumptions. Rotem and Segev [RS20] and Katz et al. [KLX20] showed that any generic algorithm for repeated squaring which is faster-than-trivial can be used to solve factoring. The result of [RS20] rules out generic constructions in the generic-ring model introduced by Aggarwal and Maurer [AM16] (see also Jager and Schwenk [JS13]). The result of [KLX20] rules out constructions in the strong algebraic group model (extending [FKL18]) wherein the adversary may use the concrete representation of group elements to make its group queries. In another work, Rotem, Segev, and Shahaf [RSS20] showed that hidden order groups are necessary for achieving "delay" functions, at least generically. The result of [RSS20] rules out generic-group constructions in Maurer's model [Mau05] (same as our proof).

1.3 Technical Overview

Throughout this overview, we use $\lambda \in \mathbb{N}$ to refer to the security parameter and let N denote the RSA modulus, where N is a product of two random λ-bit primes. We use $\mathbb{Z}_N^\star$ to denote the multiplicative group of integers mod N. We consider interactive proof systems for the repeated squaring relation $\mathcal{RS}_N$, which we represent via the function $f_{N,T}(x) = x^{2^T} \bmod N$ for any time bound $T \in \mathbb{N}$. As a warm up, we will start by considering single-round, NP-style, proof systems where the verifier is a *deterministic*, generic group algorithm. We will later show how to deal with randomized verifiers, and additionally extend to the class of *recursive* interactive proofs.

Overview of generic group proof systems. A (non-interactive) proof system consists of two parties, the prover P and the verifier V. On input a group element $x \in \mathbb{Z}_N^\star$, P's goal is to convince V that another group element y is equal to $f_{N,T}(x) = x^{2^T} \bmod N$. P is allowed to send V up to k group elements $\pi_1, \dots, \pi_k \in \mathbb{Z}_N^\star$ as well as a bit string $\mathsf{st} \in \{0,1\}^*$. Throughout the overview, we will always assume that P sends exactly k group elements as part of its proof. V processes this information and outputs 1 to accept that $y = x^{2^T} \bmod N$ or rejects otherwise. We require that the proof system satisfies the standard notions of completeness and soundness. Completeness says that if $y = x^{2^T} \bmod N$, then an honest prover P causes V to accept. We parameterize soundness by a parameter δ, which says that if $y \neq x^{2^T} \bmod N$, then no (potentially unbounded) cheating prover $P^\star$ can cause V to accept with probability more than δ.

We restrict the above model by requiring that V is a (straight-line) *generic group* verifier, whereas we still allow the prover to be unbounded and behave arbitrarily. Specifically, V takes as input the modulus N, the time bound T, the prover's string st as explicit inputs. However, V only has implicit access to the input group element x, the purported output y, and the proof elements $\pi_1, \dots, \pi_k$ sent by P. Intuitively, this means that V is allowed to multiply and divide these elements arbitrarily, as long as it does so in a way that independent of their representation. We formalize this following Maurer's generic group model [Mau05], which we outline in Section 3.

At the end of the day, we leverage the fact that V uses its explicit inputs[5] to effectively generate various exponents α, β, $\gamma_1, \dots, \gamma_k$ such that its output is given by the group element corresponding to

$$V(N, T, \mathsf{st}, x, y, \pi_1, \dots, \pi_k) = x^\alpha \cdot y^\beta \cdot \prod_{i=1}^{k} \pi_i^{\gamma_i} = g.$$

Furthermore, we can always run V with dummy elements $x, y, \pi_1, \dots, \pi_k$ and compute the exponents $(\alpha, \beta, \gamma_1, \dots, \gamma_k)$ by observing its group operations. We say that V accepts if the output group element g is equal to the multiplicative identity $1 \in \mathbb{Z}_N^\star$, and V rejects otherwise. While this

[5]If we allowed V to also use the representation of the input group elements, this would correspond to the strong algebraic group model of [KLX20].

convention may seem restrictive, as V doesn't even know whether it is accepting or rejecting, we claim that this is still very expressive as V can compute two different group elements g_1, g_2 and then output $g_1 \cdot g_2^{-1}$, which is 1 if and only if $g_1 = g_2$. Furthermore, most natural protocol for repeated squaring including [Pie19, Wes20] fall into this category.

The complexity of deterministic (NP) proofs. As a warm-up, suppose that the verifier V is deterministic. This means that for every set of explicit inputs N, T, st that V receives, it generates the same exponents $(\alpha, \beta, \gamma_1, \ldots, \gamma_k)$. Given this knowledge, we want to characterize all possible strategies a cheating prover may use. So, say a cheating prover $P^\star$ wants to fool V on any $y = x^d \neq x^{2^T} \bmod N$. Effectively, $P^\star$ can only set each group element π_i to be equal to x^{z_i} for some value z_i.[6] Then, it follows that V accepts if

$$x^\alpha \cdot x^{d \cdot \beta} \cdot \prod_{i=1}^{k} x^{z_i \cdot \gamma_i} = 1.$$

However, since the base x is shared by all of the group elements, the above holds if

$$\alpha + d \cdot \beta + \sum_{i=1}^{k} z_i \cdot \gamma_i = 0 \bmod \mathsf{Carm}(N),$$

where $\mathsf{Carm}(N)$ is Carmichael totient function, which is defined as the minimal value c such that $g^c = 1 \in \mathbb{Z}_N^\star$ for all $g \in \mathbb{Z}_N^\star$.[7] But, as long as $\vec{\gamma} = (\gamma_1, \ldots, \gamma_k) \neq \vec{0} \bmod \mathsf{Carm}(N)$, it follows that $P^\star$ can simply solve for a solution to $z_1, \ldots, z_k$ in the equation above to generate a proof that will falsely convince V that $x^d = x^{2^T}$.[8]

Still, it may be the case that V simply ignores the proof elements $\pi_1, \ldots, \pi_k$ by setting $\gamma_1, \ldots, \gamma_k = 0$. In this case, we leverage the completeness of the proof system to conclude that either V is inefficient and runs in parallel time T, or V must be able to factor N. If $y = x^{2^T} \bmod N$ and $\gamma_1, \ldots, \gamma_k = 0$, then we know, by the above equation, that V accepts if

$$\alpha + 2^T \cdot \beta = 0 \bmod \mathsf{Carm}(N).$$

We consider two different cases, either (1) $\alpha + 2^T \cdot \beta = 0 \in \mathbb{Z}$ or (2) $\alpha + 2^T \cdot \beta = c \cdot \mathsf{Carm}(N)$ for some $c \in \mathbb{Z}$.

In case (2), this actually immediately implies a probabilistic factoring algorithm for N via a well known adaptation of the Miller-Rabin primality test (formally stated in Lemma 2). Since we can compute α and β, given the code of V and the prover's string st, and hence $\alpha + 2^T \cdot \beta = c \cdot \mathsf{Carm}(N)$, this implies a factoring algorithm in the standard model given st. If the prover P is efficient, then we can compute st by ourselves, so it implies a factoring algorithm for any N, without any auxiliary advice. We emphasize, however, that it may be the case that the explicit string st sent by P helps

[6]Note that this is not true in general since $\mathbb{Z}_N^\star$ is not cyclic and hence there are group elements not represented as x^c for some $c \in \mathbb{Z}$. However, we assume this in the overview for simplicity as it captures the main idea of the proof.

[7]We note that we can simply choose x to be a group element whose order attains the maximal value $\mathsf{Carm}(N)$. This is what allows us to switch to working over the exponent without loss of generality.

[8]We note that this style of attack works for Wesolowski's (computationally sound) proof of repeated squaring [Wes20], which is an AM protocol. The adaptive root assumption essentially states that it is computationally infeasible to perform such an attack, leveraging the randomness sampled by the verifier before the prover sends its message.

V to compute some value $\alpha = 2^T \bmod \mathsf{Carm}(N)$. For example, P could have just set st to be a representation of $\mathsf{Carm}(N)$, and V simply set $\alpha = 2^T \bmod \mathsf{Carm}(N)$ and $\beta = -1$. This is why the factoring algorithm must receive the proof string st as input in general.

We split case (1) into two further subcases, either (1A) $\beta = 0$ or (1B) $\beta \neq 0$. In case (1B) where $\beta \neq 0$, this implies that

$$2^T \leq 2^T \cdot |\beta| \leq |\alpha|.$$

But that implies that V must run in parallel time T to compute x^α since $|\alpha| \geq 2^T$.

In case (1A) where $\beta = 0$ and $\alpha + 2^T \cdot \beta = 0$, it must also be the case that $\alpha = 0$. However, we've already assumed that $\gamma_1, \ldots, \gamma_k = 0$, so this means that V just always outputs 1 and accepts! So clearly, (P, V) cannot be a valid proof system as V accepts any $y \neq x^{2^T} \bmod N$ with probability 1 in this case.

In summary, if (P, V) is a sound proof system where V is a *deterministic* generic group verifier, then either:

1. V must run in parallel time at least T, or

2. there is a standard model factoring algorithm for N given the code of V and the string st output by P.

Stated another way, if V runs in parallel time less than T, then V must be able to factor N (with the help of the prover via st).

Extending to randomized verifiers. The high level outline of the lower bound for randomized verifiers is actually very similar to the case of deterministic verifiers. However, allowing the verifier to use randomness to determine its exponents introduces many highly non-trivial challenges. The key distinction between deterministic and randomized verifiers is that randomized verifiers are allowed choose their exponents as a function of their randomness, so the attack where a cheating prover simply solves a single equation to fool the verifier no longer works. Instead, the cheating prover needs to satisfy a random equation with better than δ probability in order to violate soundness. Still, we will show how we can use the verifier's exponents to factor, or argue that the verifier must have parallel running time greater than $T/(k+1)$ with high probability.

Throughout, we will consider a fixed set of explicit inputs N, T, and st received by the verifier. Then, for any random string $\rho \in \{0,1\}^\lambda$ sampled by the verifier, we use $\mathsf{coef}(\rho)$ to denote the exponents that V uses to compute its output. So, if

$$V(N, T, \mathsf{st}, x, y, \pi_1, \ldots, \pi_k; \rho) = x^\alpha \cdot y^\beta \cdot \prod_{i=1}^{k} \pi_i^{\gamma_i},$$

then we say that $\mathsf{coef}(\rho) = (\alpha, \beta, \gamma_1, \ldots, \gamma_k)$. We note that we refer to these exponents as "coefficients" as they will correspond to coefficients in a system of equations over the exponent, hence the notation $\mathsf{coef}(\rho)$.

Our main strategy is to sample many different values $\rho_1, \ldots, \rho_n$ such that $||\mathsf{coef}(\rho_i)||_{\max} << 2^{T/(k+1)}$ for each $i \in [n]$, where $|| \cdot ||_{\max}$ indicates the maximum absolute value in the coefficient vector. If this isn't possible, then that means that the verifier must run in parallel time at least $T/(k+1)$, and we are done. Otherwise, it remains to show that we can either use these coefficients to factor or show that (P, V) is not a valid proof system. For each randomness value ρ_i, let

$\mathsf{coef}(\rho_i) = (\alpha_i, \beta_i, \gamma_{i,1}, \ldots, \gamma_{i,k})$ denote the corresponding coefficient vector for ρ_i. We combine all of these coefficients together in the following way. Let $\Gamma \in \mathbb{Z}^{n\times k}$ be the matrix consisting of all of the $\gamma_{i,j}$ values, and let $\vec{\alpha}, \vec{\beta} \in \mathbb{Z}^n$ be vectors of the α_i and β_i values. A key property we will leverage is that the system of equations $\Gamma \cdot \vec{z} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \mathsf{Carm}(N)$ has a solution for $d = 2^T$ by completeness, but does not have a solution for any $d \neq 2^T \bmod \mathsf{Carm}(N)$ by soundness (with high probability), which we explain next.

For simplicity, we will assume throughout this overview that the proof elements π_j potentially output by the prover are all equal to x^{z_j} for some $z_j \in \mathbb{Z}$. Then, for $y = x^{2^T}$ and all $i \in [n]$, completeness tells us that there must be a solution for $z_1, \ldots, z_k$ to the equation

$$\alpha_i + 2^T \cdot \beta_i + \sum_{j=1}^{k} \gamma_{i,j} \cdot z_j = 0 \bmod \mathsf{Carm}(N).$$

Since the prover's proof must work for all randomness values by completeness, we know that the prover's vector $\vec{z} = (z_1, \ldots, z_k)^\top$ actually satisfies

$$\Gamma \cdot \vec{z} = -\vec{\alpha} - 2^T \cdot \vec{\beta} \bmod \mathsf{Carm}(N).$$

However, for any $d \neq 2^T \bmod \mathsf{Carm}(N)$ corresponding to $x^d \neq x^{2^T}$, we use soundness to show that

$$\nexists \vec{z},\ \Gamma \cdot \vec{z} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \mathsf{Carm}(N),$$

as long as we sample enough vectors n. At a very high level, this follows since each newly sampled coefficient vector must restrict the space of solutions in a non-trivial way, since otherwise the same solution will work with good probability for many different choices of exponents. So we set n large enough such that, with high probability, the space of possible solutions for any $d \neq 2^T \bmod \mathsf{Carm}(N)$ is empty. The details of this argument are given in Section 4.1.

Next, we prove a key technical lemma that allows us to relate whether or not a system of equations mod $\mathsf{Carm}(N)$ has a solution. Specifically, we show that there exists an efficiently computable matrix M that satisfies the following two properties:

1. If there exists a solution $\vec{z}$ such that $\Gamma \cdot \vec{z} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \mathsf{Carm}(N)$, then $M \cdot (-\vec{\alpha} - d \cdot \vec{\beta}) = \vec{0} \bmod \mathsf{Carm}(N)$.
2. If $M \cdot (-\vec{\alpha} - d \cdot \vec{\beta}) = \vec{0}$ over $\mathbb{Z}$, then there exists a solution $\vec{z}$ such that $\Gamma \cdot \vec{z} = (-\vec{\alpha} - d \cdot \vec{\beta})$ over $\mathbb{Z}$ (and hence $\bmod \mathsf{Carm}(N)$).

Furthermore, we show that $||M \cdot \vec{v}||_{\max} < 2^T$ when $||\vec{v}||_{\max}, ||\Gamma||_{\max} << 2^{T/(k+1)}$. When working over a field, such a result is well known by simply converting Γ into reduced row echelon form and the linear function M is closely related to the determinant of Γ. However, working over the integers mod $\mathsf{Carm}(N)$, this becomes much messier to work with. At a very high level, we show the lemma by first converting Γ to its Hermite normal form H, which is the integer counterpart to reduced row echelon form. We then augment the matrix H with the column $(-\vec{\alpha} - d \cdot \vec{\beta})$ and apply linear operations to zero out the last column to construct the matrix M. However, working over the integers, we must be careful to make sure that the values don't blow up in order to get our desired bound on $||M \cdot \vec{v}||_{\max}$. The full details for the proof of this technical lemma are provided in Section 4.2.

Armed with our key technical lemma and the observations above, we are ready to complete the logic of our result, which follows the same high level structure as the deterministic case. Given M, we compute $\vec{v} = M \cdot (-\vec{\alpha} - 2^T \cdot \vec{\beta})$. By completeness, we know that there exists a vector $\vec{z}$ such that $\Gamma \cdot \vec{z} = (-\vec{\alpha} - d \cdot \vec{\beta}) \bmod \mathsf{Carm}(N)$, so by the technical lemma, we know that $\vec{v} = \vec{0} \bmod \mathsf{Carm}(N)$. We consider two different cases, either (1) $\vec{v} = \vec{0}$ over $\mathbb{Z}$ or (2) there exists an index i such that $\vec{v}_i = c \cdot \mathsf{Carm}(N)$ for $c \in \mathbb{Z}$. In case (2), we can factor given $\vec{v}_i$ using the variant of the Miller-Rabin primality test, so we are done.

For case (1), we use the fact that M is linear, so

$$\vec{v} = M \cdot (-\vec{\alpha} - 2^T \cdot \vec{\beta}) = -M \cdot \vec{\alpha} - 2^T \cdot M \cdot \vec{\beta} \bmod \mathsf{Carm}(N).$$

We consider two further subcases, either (1A) $M \cdot \vec{\beta} = \vec{0}$ over $\mathbb{Z}$ or (1B) there exists an index i such that $M_i \cdot \vec{\beta} \neq 0$. In case (1B), this implies that

$$2^T \leq 2^T \cdot |M_i \cdot \vec{\beta}| \leq |M_i \cdot \vec{\alpha}|,$$

but we show in our key technical lemma that $|M_i \cdot \vec{\alpha}| < 2^T$. So case (1B) cannot happen.

In case (1A) where $M \cdot \vec{\beta} = \vec{0}$, this actually implies that $M \cdot \vec{\alpha} = \vec{0}$ since we have already assumed that $\vec{v} = M \cdot (-\vec{\alpha} - 2^T \cdot \vec{\beta}) = \vec{0}$. But, this implies that $M \cdot (-\vec{\alpha} - d \cdot \vec{\beta}) = \vec{0}$ over $\mathbb{Z}$ for any $d \neq 2^T \bmod \mathsf{Carm}(N)$! So, by our key technical lemma, we conclude that there exists a solution over $\mathbb{Z}$, and hence mod $\mathsf{Carm}(N)$ for some $d \neq 2^T \bmod \mathsf{Carm}(N)$. However, we argued above that this cannot be the case by soundness (with high probability).

Combining the above, we've ruled out the possibility of case (1), so case (2) must hold, which implies we can factor with high probability. So, in summary, if (P, V) is a sound proof system where V is now a *randomized* generic group verifier and P sends at most k group elements in its proof, then either:

1. V runs in parallel time at least $T/(k+1)$ with high probability, or

2. there is a standard model factoring algorithm for N given the code of V and the string st output by P.

An alternative way to view this result is as follows. If V run in parallel time less than $T/(k+1)$ with good probability, then either it must "know" a factorization of N to be able to reduce its exponents mod $\mathsf{Carm}(N)$, or there must be a cheating strategy that falsely convinces V on such randomness values. Hence, if you want both statistical security and an efficient verifier V, it must be the case that V can factor N.

Recursive interactive proofs. We next discuss how our result for one-round, MA-style, proofs extends to the class of recursive interactive proofs. First, we define what we mean by a r-round recursive interactive proof for the function $f_{N,T}(x) = x^{2^T} \bmod N$. In each round i, there is an input statement (x, y, T) claiming that $y = x^{2^T} \bmod N$. P starts the round by sending a string $\mathsf{st} \in \{0,1\}^*$ and up to k group elements $\pi_1, \ldots, \pi_k$. V then responds with a random string $\rho \leftarrow \{0,1\}^\lambda$. If i is the last round, V uses its randomness ρ and the message from P to decide whether or not $y = x^{2^T} \bmod N$. Otherwise, P and V both use a generic group algorithm A_i to compute a new statement (x', y', T') given the prover's message and the verifier's random coins, and they start a new independent (recursive) proof for this statement with one fewer round.

The overall running time of V is simply the running time of A_i in each round i, plus its final running time to compute its output at the end of the protocol. In addition to standard notions of completeness and soundness, we require that if (x, y, T) is valid at the beginning of the round, then (x', y', T') is also valid for the start of the next round. However, if (x, y, T) starts as invalid, so $y \neq x^{2^T} \bmod N$, then we require that (x', y', T') is invalid with probability at least $1 - \delta$.

Due to the recursive nature of this interactive proof, we are able to reduce to the one-round case to show that in each round T' cannot shrink too much relative to T, assuming A_i (and hence V) runs in low parallel time. If there exists a round i such that T' is much smaller than T, then we could construct a proof system $(\widehat{P}, \widehat{V})$ for $y = x^{2^T} \bmod N$ as follows. The prover $\widehat{P}$ sends whatever P would have sent in round i. Then, $\widehat{V}$ runs A_i to compute (x', y', T') and outputs $(x')^{2^{T'}} \cdot (y')^{-1}$. It follows that $\widehat{V}$ runs in time corresponding to the running time of A_i plus T', which is dominated by T'. By our result for one-round proofs, this means that T' must be at least $T/(k+1)$ with high probability, otherwise we can construct a factoring algorithm given the proof string st from P in round i. Hence, after $r - 1$ rounds, the final time bound T' must be at least $T/(k+1)^{r-1}$ and V must run in parallel time at least $T/(k+1)^r$ to be a valid proof system.

In summary, if (P, V) is a *recursive*, generic group, r-round interactive proof for $f_{N,T}(x) = x^{2^T} \bmod N$, where the prover sends at most k group elements per round, then either:

1. V runs in parallel time at least $T/(k+1)^r$ with high probability, or
2. there is a standard model factoring algorithm for N given the code of V and the transcript generated by an honest prover P.

We make note that in our formal result, we actually only get a bound of roughly $(1/2^r) \cdot (T/(k+1)^r)$. This is because we actually lose a factor of 2 in the analysis of our one-round bound, and this constant factor gets amplified over many rounds in the interactive setting. We leave it as an open question whether or not this can be removed in a tighter analysis.

2 Preliminaries

For any $n \in \mathbb{N}$, we use $[n] = \{1, \ldots, n\}$ to denote the set from 1 to n. For a distribution X, we denote by $x \leftarrow X$ the process of sampling a value x from the distribtion X. For a set $\mathcal{X}$, we use $x \leftarrow \mathcal{X}$ to denote the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For a bit string $\mathsf{st} \in \{0,1\}^*$, we use $|\mathsf{st}|$ to denote the length of st. Throughout, we use $\lambda \in \mathbb{N}$ to denote the security parameter.

2.1 Number Theory

In this work, we consider the multiplicative group of integers mod N, denoted by $\mathbb{Z}_N^\star$, where N is a product of two primes. Specifically, for any $\lambda \in \mathbb{N}$, we let $\mathsf{ModGen}(1^\lambda)$ denote the algorithm that samples two random primes p, q in the interval $[2^\lambda, 2^{\lambda+1})$ and outputs $N = p \cdot q$. The group is given by $\mathbb{Z}_N^\star = \{x \in [1, N) : \gcd(x, N) = 1\}$, and multiplication in the group corresponds to multiplication over $\mathbb{Z}$ mod N. When it is clear from context we are working in the group $\mathbb{Z}_N^\star$, we will omit mod N when discussing multiplication of group elements.

The main language we consider in this work is the repeated squaring relation, $\mathcal{RS}_N$, defined as follows

$$\mathcal{RS}_N = \left\{(x, y, T) \mid y = x^{2^T} \bmod N\right\}.$$

For a particular value of N and T, we represent this relation by the function $f_{N,T}(x) = x^{2^T} \bmod N$. It is widely believed that $f_{N,T}$ cannot be computed and $\mathcal{RS}_N$ cannot be decided in depth less than T even with $\mathsf{poly}(\lambda, T)$ parallel processors. We focus on the proof complexity of this language in this work.

For any $a, b \in \mathbb{Z}$, we use $\gcd(a, b)$ and $\operatorname{lcm}(a, b)$ to denote the greatest common divisor and least common multiple of a and b, respectively. Specifically, $\gcd(a, b)$ is the maximal $c \in \mathbb{N}$ such that c divides a and b, and lcm is the minimal $c \in \mathbb{N}$ such that a and b both divide c. Let $a, b \in \mathbb{Z}$, then there always exist integers c, d such that $c \cdot a + d \cdot b = \gcd(a, b)$. c and d are known as Bezout coefficients for a and b. While Bezout coefficients may not be unique, we note that there always exist bezout coefficients such that $|c|, |d| \leq \max(|a|, |b|)$, and these are the coefficients given by the standard euclidean algorithm.

We denote by $\varphi(N) = |\mathbb{Z}_N^\star|$, known as the Euler totient function of N, and $\mathsf{Carm}(N) = \min\{a \in \mathbb{N} : \forall g \in \mathbb{Z}_N^\star, g^a = 1\}$, known as the Carmichael totient function. For $\lambda \in \mathbb{N}$ and $N \in \operatorname{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$ such that $N = p \cdot q$, it holds that

$$\varphi(N) = (p-1) \cdot (q-1), \text{ and } \mathsf{Carm}(N) = \operatorname{lcm}(p-1, q-1).$$

For a specific element $g \in \mathbb{Z}_N^\star$, we define the order of g, $\mathsf{ord}(g)$, to be the minimum $c \in \mathbb{N}$ such that $g^c = 1 \in \mathbb{Z}_N^\star$.

In this work, we use the fact that for $N = p \cdot q$, $\mathbb{Z}_N^\star \cong \mathbb{Z}_p^\star \times \mathbb{Z}_q^\star$, where $\mathbb{Z}_p^\star$ and $\mathbb{Z}_q^\star$ are each cyclic groups of order $\varphi(p) = p - 1$ and $\varphi(q) = q - 1$, respectively. Let g_p and g_q be generators for the corresponding subgroups. Then, we can write any group element $h \in \mathbb{Z}_N^\star$ in the form $h = g_p^a \cdot g_q^b$ for some $a, b \in \mathbb{N}$. For convenience of notation, we will use $h|_p$ to denote the p "component" of h and $h|_q$ to denote the q component, so $a = h|_p$ and $b = h|_q$ above.

In order to translate between results mod a composite number Φ and its solutions mod its prime power divisors, we make use of the Chinese remainder theorem (CRT). We use the following version of CRT.

Lemma 1. *Let $k \in \mathbb{N}$, $n_1, \ldots, n_k, a_1, \ldots, a_k \in \mathbb{N}$. Then, the set of equations*

$$x = a_i \bmod n_i$$

has a solution over $\mathbb{Z}$ if and only if for all $i, j \in [k]$, $a_i = a_j \bmod \gcd(n_i, n_j)$. Moreover, any two solutions x_1, x_2 satisfy $x_1 = x_2 \bmod \operatorname{lcm}(n_1, \ldots, n_k)$.

The following lemma, based on the Miller-Rabin primality test [Mil76, Rab80], gives a probabilistic factoring algorithm given any non-zero multiple of $\mathsf{Carm}(N)$. For the proof of the lemma and further discussion, we refer the reader to Section 10.4 of Shoup [Sho06].

Lemma 2 (Factoring Lemma). *Let $\lambda \in \mathbb{N}$, $N \in \operatorname{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$, and $m = c \cdot \mathsf{Carm}(N)$ for $c \in \mathbb{Z}$ such that $c \neq 0$. For any $\delta \colon \mathbb{N} \to [0, 1]$, there exists a probabilistic algorithm A that runs in $\mathsf{poly}(\lambda, \log(1/\delta(\lambda)))$ time such that*

$$\Pr\left[p, q \leftarrow A(1^\lambda, N, m) : N = p \cdot q\right] \geq 1 - \delta(\lambda).$$

2.2 Linear Algebra

Let M be a matrix in $\mathbb{Z}^{m\times\ell}$. For $i \in [m]$, $j \in [\ell]$, we use M_i to denote the ith row and $M_{i,j}$ to denote the element in the ith row and jth column. We use $M^\top$ to denote the transpose of a matrix. We treat vectors $\vec{v} \in \mathbb{Z}^n$ as column vectors, so implicitly of the form $\vec{v} \in \mathbb{Z}^{n\times 1}$. To take the dot product of two vectors $\vec{v}, \vec{u}$, we write $\vec{v}^\top \cdot \vec{u}$. If $v \in \mathbb{Z}^{m\times 1}$ is a vector, we simply write v_i to denote the ith component. We write $||M||_{\max} = \max_{i\in[m],j\in[\ell]} |M_{i,j}|$ to denote the largest element in absolute value in the matrix M. For a matrix $M^{(1)} \in \mathbb{Z}^{m\times\ell_1}$ and a matrix $M^{(2)} \in \mathbb{Z}^{m\times\ell_2}$, we write $M' = (M^{(1)}|M^{(2)})$ to denote the augmented matrix which appends $M^{(2)}$ to the right of $M^{(1)}$ to get the matrix $M' \in \mathbb{Z}^{m\times(\ell_1+\ell_2)}$.

For any composite Φ, let $\mathbb{Z}_\Phi$ be the ring of integers mod Φ. We say that a function $f\colon \mathbb{Z}_\Phi^n \to \mathbb{Z}_\Phi^n$ is linear if for any vectors $\vec{g}, \vec{h} \in \mathbb{Z}_\Phi$ and $a, b \in \mathbb{Z}$, it satisfies $f(a\cdot\vec{g} + b\cdot\vec{h}) = a\cdot f(\vec{g}) + b\cdot f(\vec{h})$. For the purpose of this work, it suffices to say that a function is linear if there exists some matrix M such that $f(\vec{g}) = M\cdot\vec{g}'$ where $\vec{g}'$ is equal to $\vec{g}$ appended by 1.

Let $\mathsf{Perm}(n)$ denote the set of all permutations over $[n]$. For a permutation $\sigma \in \mathsf{Perm}(n)$, we write $\mathsf{sign}(\sigma)$ to denote the sign of σ, i.e. 1 if there are an even number transpositions from the identity to σ, and -1 otherwise. For a square matrix M, the determinant of M is given by $\det(M) = \sum_{\sigma\in\mathsf{Perm}(n)} \mathsf{sign}(\sigma)\cdot\prod_{i=1}^n M_{i,\sigma(i)}$. It follows by definition of the determinant that $\det(M) \le n!\cdot||M||_{\max}^n$. We say that an integer matrix $U \in \mathbb{Z}^{m\times m}$ is unimodular if $\det(U) \in \{+1, -1\}$.

Let $\vec{v}^{(1)}, \ldots, \vec{v}^{(n)} \in \mathbb{Z}^m$ be a set of vectors. This determines a lattice

$$\mathcal{L} = \mathcal{L}\left(\vec{v}^{(1)}, \ldots, \vec{v}^{(n)}\right) = \left\{\sum_{i=1}^m c_i\cdot\vec{v}^{(i)} : c_1, \ldots, c_m \in \mathbb{Z}\right\}$$

of points spanned by these vectors. For a lattice $\mathcal{L}$, we refer to a basis of the lattice as a set of vectors $\vec{b}^{(1)}, \ldots, \vec{b}^{(m)}$, often written in matrix matrix $B = (\vec{b}^{(1)}|\ldots|\vec{b}^{(m)})$, that are linearly independent over $\mathbb{R}$ and $\mathcal{L} = \mathcal{L}(B)$. A lattice is unique up to multiplication of B by a unimodular matrix U, so when the basis is clear from context, we refer simply to the lattice $\mathcal{L}$. The determinant of a lattice $\det(\mathcal{L})$ is defined to be the volume of the parallelepiped formed by a set of basis vectors over $\mathbb{R}^m$. By Hadamard's inequality, it is known that

$$\det(\mathcal{L}(B)) \le ||B||_{\max}^m \cdot m^{m/2}.$$

We next define the Hermite normal form (HNF) of an integer matrix $M \in \mathbb{Z}^{m\times n}$. We use the notion of column-style HNF, defined via right multiplication by a unimodular matrix, in contrast to row-style HNF.

Definition 1 (Hermite Normal Form)**.** A matrix $H \in \mathbb{Z}^{m\times n}$ is in Hermite normal form if the following hold:

1. Lower triangular: For some $h \le n$, there exists a sequence $1 \le i_1 < i_2 < \ldots < i_h \le n$ such that $H_{i,j} \ne 0 \Rightarrow i > i_j$.

2. Row-reduced: For all $k \le j \le n$, $0 \le H_{i_j,k} \le H_{i_j,j}$.

We additionally use the fact that the HNF of a matrix $M \in \mathbb{Z}^{m\times n}$ has entries bounded by $||M||_{\max}^n$. See [KB14] for a proof of this claim.

When working over a field $\mathbb{F}$, such as the integers mod a prime p or the rationals $\mathbb{Q}$, we can define standard notions like span and rank. The span of a set over vectors over an n dimensional vector space over a field $\mathbb{F}$ is defined as the set of all linear combinations of the vectors, with coefficients from the field $\mathbb{F}$. When clear from context, we use span in the context of integers to refer to the set of linear combinations with coefficients from $\mathbb{Z}$, as in the definition of a lattice. The rank of a matrix or vector space over a field $\mathbb{F}$ is the size of the minimal set of vectors that spans the space over $\mathbb{F}$.

2.3 Concentration Inequalities

Concentration inequalities allow us to bound the probability that certain random variables take values too far away from their mean. In this work, we use the following version of the well known Chernoff-Hoeffding bound [Hoe63].

Lemma 3 (Chernoff-Hoeffding Bound [Hoe63])**.** *Let $X = \sum_{i=1}^{m} X_i$ such that $X_i \in [0,1]$ are independent random variables. Let $\mu = \mathrm{E}[X]$. Then, for all t,*

$$\Pr[|X - \mu| > t] \leq 2e^{-2t^2/m}.$$

3 Generic Group Proof Systems

We next give the details for the generic group model we use in this work. Then we define proof systems where the verifier is restricted to generic group operations.

3.1 The Generic Group Model

In this work, we use Maurer's generic group model abstraction [Mau05], following the related works of Aggarwal and Maurer [AM16] and Rotem and Segev [RS20]. We note that this is not the same as Shoup's random representation model [Sho97]. See the work of Zhandry [Zha22] for a detailed comparison between these two models.

Informally, a generic group algorithm is one that can perform arbitrary group operations as long as the operations performed are independent of the representation of the group elements. At a high level, we model this by giving the algorithm indirect access to its input group elements via pointers into a table, and each new multiplication or division adds a new element to the table and returns the corresponding pointer.

Formally, we consider the multiplicative group $\mathbb{Z}_N^\star$ in this work, where N is an RSA modulus in $\mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$ for some security parameter $\lambda \in \mathbb{N}$. A generic group algorithm A receives N as input as an explicit bit string and also receives access to a table Table via an oracle $\mathcal{O}$ that stores the group elements computed so far. Initially, Table contains the identity $v_0 = 1 \in \mathbb{Z}_N^\star$ at index 0, and all of the group elements $x_1, \ldots, x_k \in \mathbb{Z}_N^\star$ provided as input to A in indices $1, \ldots, k$. A can make queries to the oracle $\mathcal{O}$ via the following syntax:

- Mutliplication: On input $(i_1, i_2, j, \times)$, the oracle $\mathcal{O}$ checks that the values v_{i_1} and v_{i_2} at indices i_1 and i_2 in Table are non-empty and not $\perp$. If so, $\mathcal{O}$ computes $v_{i_1} \circ v_{i_2}$ and stores the result at index j in Table. Otherwise, $\mathcal{O}$ stores $\perp$ at index j.

- Division: On input $(i_1, i_2, j, \div)$, the oracle $\mathcal{O}$ additionally checks v_{i_2} is invertible. If so, $\mathcal{O}$ computes $v_{i_2}^{-1}$ and stores $v_i \times v_{i_2}^{-1}$ at index j in Table, if applicable. Otherwise, $\mathcal{O}$ stores $\perp$ at index j.

We note that Maurer's generic group model usually includes equality queries, which we do not handle in this work. An algorithm A that does not issue any equality queries is known as a *straight-line* algorithm, so for this reason, we state our formal results for straight-line generic group algorithms to avoid confusion. We note that generic-ring algorithms are defined similarly as above, but they also include addition and subtraction queries with essentially the same syntax.

For a group element g computed by A, we use $\widehat{g}$ to denote the pointer to the corresponding element g in the table Table. We abuse notation slightly and whenever we write that A receives a group element g as input, we mean that it receives a pointer $\widehat{g}$ to the element in the corresponding table Table.

We allow generic group algorithms to receive and output both "explicit" values, represented by bit strings, and "implicit" values indicating group elements, represented by pointers into Table. We can think of all of the explicit values as helping the generic algorithm decide how to invoke the oracle $\mathcal{O}$ to perform generic operations.

A randomized generic group algorithm also receives as input a string $\rho \in \{0,1\}^\lambda$ (we assume λ bits of randomness for simplicity, however this could be extended arbitrarily). For any input inp, We denote $A(1^\lambda, N, \mathsf{inp}; \rho)$ the randomized generic group algorithm with random tape ρ.

Measuring complexity. Let A be a generic group algorithm. We denote by $\mathsf{Time}_A(1^\lambda, N, \mathsf{inp}; \rho)$ the total running time of A on the given inputs with random tape ρ, where each oracle query costs a single unit of time. Additionally, we allow A to be a parallel algorithm. Following Rotem and Segev [RS20], we model parallel generic group algorithms A by allowing A to issue oracle queries in "rounds". In each round, A can issue any number of oracle queries to $\mathcal{O}$ in a single time step via multiple processors. We use $\mathsf{Width}_A(1^\lambda, N, \mathsf{inp}; \rho)$ to denote the maximum number of processors used by A at any time step and $\mathsf{ParTime}_A(1^\lambda, N, \vec{x}; \rho)$ to denote the number of sequential time steps that it takes for A to compute its output. Whenever we omit input/ randomness parameters from Time_A, Width_A, or $\mathsf{ParTime}_A$, we mean the worst case running time over an arbitrary choice of input parameters.

The behavior of generic group algorithms. Let $\lambda \in \mathbb{N}$ and $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$. Let A be a straight-line generic group algorithm such that $A(1^\lambda, N, \mathsf{st}, \vec{x}; \rho)$ takes as input an explicit string $\mathsf{st} \in \{0,1\}^*$ and group elements $\vec{x} = x_1, \ldots, x_k \in \mathbb{Z}_N^\star$ and outputs a group element g. As A is only allowed to perform generic operations, it follows that A's output is of the form $\prod_{i=1}^k x_i^{\gamma_i}$ for $\gamma_1, \ldots, \gamma_k \in \mathbb{Z}$. Furthermore, by running A, we can compute these coefficients by providing arbitrary pointers as input to A in place of $\vec{x}$. We use the notation $\mathsf{coef}_{V,\lambda,N,\mathsf{st}}(\rho) = (\gamma_1, \ldots, \gamma_k)^\top$ to denote the coefficient vector of V on input ρ for security parameter λ, modulus N, and explicit string st. We note that the main distinction between our model and the strong algebraic group model of [KLX20] is that they allow the coefficient vector to additionally depend on the bit representations of the input group elements.

Relating parallel running time to degree. Its easy to see that a straight-line generic group algorithm that computes $A(1^\lambda, N, \mathsf{st}, \vec{x}; \rho) = \prod_{i=1}^k x_i^{\gamma_i}$, where $\gamma_1, \ldots, \gamma_k$ are given by $\vec{\gamma} = \mathsf{coef}_{A,\lambda,N,\mathsf{st}}(\rho)$, must run in depth at least $\log ||\vec{\gamma}||_{\max}$. This can be shown by induction for $||\vec{\gamma}||_{\max}$ equal to 2^i for $i \geq 0$. If $||\vec{\gamma}||_{\max} = 2^0 = 1$, then it may be the case that A just immediately outputs a group element in 0 steps, satisfying the base case. Suppose that $||\vec{\gamma}||_{\max} = 2^i$. After $i-1$ steps, the maximum exponent in absolute value of any group element in Table is 2^{i-1} by assumption. So, in the next time step, A can issue a multiplication query multiplying two such elements together. However, this will result in an element with depth at most 2^i, as required. It follows that

$$\mathsf{ParTime}_A(1^\lambda, N, \mathsf{st}, \vec{x}; \rho) \geq \log ||\mathsf{coef}_{A,\lambda,N,\mathsf{st}}(\rho)||_{\max}$$

for all $\lambda \in \mathbb{N}$, $N \in \operatorname{Supp}\big(\mathsf{ModGen}(1^\lambda)\big)$, string $\mathsf{st} \in \{0,1\}^*$, input elements $\vec{x}$, and random string $\rho \in \{0,1\}^\lambda$.

We additionally note that, even if we only require A to compute a high degree function with high probability and with pre-processing over a random input, then the same lower bound holds by the work of Rotem and Segev [RS20].

3.2 Proof Systems in the Generic Group Model

A proof system consists of two algorithms: the prover P and the verifier V. For a language L, P and V interact on common input x over potentially many rounds until V either accepts or rejects. In order to be non-trivial, the prover P must have some additional capabilities compared to the verifier V. For classical proof systems, the prover P is an unbounded algorithm while V is polynomially bounded. The two main properties of a proof system are completeness and soundness. Completeness stipulates that P convinces V on $x \in L$, and δ-soundness stipulates no cheating prover $P^\star$ can convince V on $x \notin L$ with probability better than δ.

We consider *generic group* proof systems for languages defined by a function f defined over a group $\mathbb{Z}_N^\star$ for $\lambda \in \mathbb{N}$ and $N \in \operatorname{Supp}\big(\mathsf{ModGen}(1^\lambda)\big)$. For such proof systems, we restrict V to be a generic group algorithm that makes a bounded number of group multiplication and division queries, whereas P may still be an unbounded (not necessarily generic) algorithm that sends a bit string and group elements to V. So, for a function f, P and V receive an input a security parameter 1^λ, the group description N, an input group element x, and the output of the function $f(x)$ as common input. P sends a bit string $\mathsf{st} \in \{0,1\}^*$ and sequence of group elements $\pi_1, \ldots, \pi_k$ to V, which V receives access to via pointers into a table as a generic group algorithm. V then performs generic computations and outputs a pointer to a group element $\widehat{g}$ and "accepts" if the corresponding group element $g = 1$.

Definition 2 (Generic Group Proof Systems)**.** Let $\delta \colon \mathbb{N} \to [0,1]$ and $k \colon \mathbb{N} \to \mathbb{N}$. For any $\lambda \in \mathbb{N}$, $N \in \operatorname{Supp}\big(\mathsf{ModGen}(1^\lambda)\big)$, let $f \colon \mathbb{Z}_N^\star \to \mathbb{Z}_N^\star$ be a function. We say that the pair (P, V) is a k-element generic group proof system for f with δ-soundness if V is a generic group algorithm, and for all $\lambda \in \mathbb{N}$, $N \in \operatorname{Supp}\big(\mathsf{ModGen}(1^\lambda)\big)$, and $k = k(\lambda)$, the following hold:

- Completeness: For all $x \in \mathbb{Z}_N^\star$, let $\mathsf{st}, \pi_1, \ldots, \pi_k$ be the output of $P(1^\lambda, N, x, f(x))$, then it holds that
$$V(1^\lambda, N, \mathsf{st}, x, f(x), \pi_1, \ldots, \pi_k) = 1.$$
- Soundness: For all $x \in \mathbb{Z}_N^\star$, $y \neq f(x)$, and algorithms $P^\star$ such that $P^\star(1^\lambda, N, x, y)$ outputs a string st and group elements $z_1, \ldots, z_k$, it holds that
$$\Pr_{\rho \leftarrow \{0,1\}^\lambda}\Big[V(1^\lambda, N, \mathsf{st}, x, y, z_1, \ldots, z_k) = 1\Big] \leq \delta(\lambda).$$

If the verifier V is a straight-line algorithm, we say that (P, V) is a straight-line generic group proof system.

4 One Round Proofs

In this section, we prove our main theorem. Let $\lambda \in \mathbb{N}$ and $N \in \text{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$. We show that if there is a generic group proof system with a straight-line verifier that runs in parallel time less than $T/2(k+1)$ with probability ϵ, then there is a $\mathsf{poly}(1/\epsilon) \cdot \mathsf{Time}_V$ algorithm that factors N. We define some useful notation for the theorem first, and then provide a high level outline of the proof structure.

For each randomness ρ, let $\mathsf{coef}_{V,\lambda,N,\mathsf{st}}(\rho) = (\gamma_1, \ldots, \gamma_k, \alpha, \beta)^\top$ be the coefficients such that $V(1^\lambda, N, \mathsf{st}, x, y, z_1, \ldots, z_k)$ outputs $x^\alpha \cdot y^\beta \cdot \prod_{i=1}^k z_i^{\gamma_i}$. As V is a generic group algorithm, we can compute $\mathsf{coef}_{V,\lambda,N,\mathsf{st}}(\rho)$ by simply running $V(1^\lambda, N, \mathsf{st}, x, y, z_1, \ldots, z_k)$ for generic elements $x, y, z_1, \ldots, z_k$ and keep track of the operations of V. For notational convenience, when $V, \lambda, N, \mathsf{st}$ are clear from context, we simply write $\mathsf{coef}(\rho)$. We also define $\mathsf{dcoef}_{V,\lambda,N,\mathsf{st}}(\rho, d)$ to denote the vector $(\gamma_1, \ldots, \gamma_k, \alpha + d \cdot \beta)^\top$, where $(\gamma_1, \ldots, \gamma_k, \alpha, \beta)$ are given by $\mathsf{coef}(\rho)$, which will be useful in our analysis.

For simplicity of presentation, the following theorem contains the core argument in the proof. We have two key cases that come up in the analysis. We analyze case (1A) in Section 4.1 and case (1B) in Section 4.2. We additionally provide proofs of additional claims needed after the proof of the theorem.

Theorem 3. *Let $\lambda \geq 2, T \in \mathbb{N}$, $k\colon \mathbb{N} \to \mathbb{N}$, $\delta, \epsilon\colon \mathbb{N} \to [0,1]$, $N \in \text{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$, and (P, V) be a k-element straight-line generic-group proof system for the function $f_{N,T}(x) = x^{2^T} \bmod N$ with soundness error δ. For any $(\mathsf{st}, \pi_1, \ldots, \pi_{k(\lambda)}) \in \text{Supp}\left(P(1^\lambda, N, T, x, f_{N,T}(x))\right)$. If*

$$\Pr_\rho\left[\mathsf{ParTime}_V(1^\lambda, N, T, \mathsf{st}) < \frac{T}{2(k(\lambda)+1)} - \log(2k(\lambda))\right] \geq \max(2\delta(\lambda), \epsilon(\lambda)),$$

then there exists a standard model probabilistic $\mathsf{poly}(\lambda, k(\lambda), T, 1/\epsilon(\lambda)) \cdot \mathsf{Time}_V(1^\lambda, N, \mathsf{st})$ time algorithm A such that

$$\Pr\left[p, q \leftarrow A\left(1^\lambda, N, k, T, \mathsf{st}, 1/\epsilon(\lambda)\right) : N = p \cdot q\right] \geq 1 - 2^{-\lambda}.$$

Proof. In order to prove the theorem, we show the existence of a factoring algorithm assuming

$$\Pr_\rho\left[||\mathsf{coef}_{V,\lambda,N,\mathsf{st}}(\rho)||_{\max} < \frac{1}{2k(\lambda)} \cdot 2^{T/2(k(\lambda)+1)}\right] \geq \max(2\delta(\lambda), \epsilon(\lambda)).$$

This implies the theorem since $\mathsf{ParTime}_V(1^\lambda, N) < T/2(k+1) - \log(2k)$ implies $||\mathsf{coef}_{V,\lambda,N,\mathsf{st}}(\rho)||_{\max} < \frac{1}{2k} \cdot 2^{T/2(k+1)}$.

Fix any $\lambda \in \mathbb{N}$, and for notational convenience we define $k = k(\lambda)$, $\epsilon = \epsilon(\lambda)$, and $\delta = \delta(\lambda)$. For any $T \in \mathbb{N}$ and $N \in \text{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$, we construct a factoring algorithm $A(1^\lambda, N, k, T, \mathsf{st}, 1/\epsilon)$ as follows.

1. Let $n = 16\lambda^3(k+1)$. Sample $2n\lambda/\epsilon^2$ coefficient vectors, and let $\rho_1, \ldots, \rho_n$ be vectors such that $||\mathsf{coef}(\rho_i)||_{\max} < \frac{1}{2k} \cdot 2^{T/2(k+1)}$ for all $i \in [n]$. If there are fewer than n such vectors, abort and output $\perp$. Otherwise, for each ρ_i, let $\mathsf{coef}(\rho_i) = (\gamma_{i,1}, \ldots, \gamma_{i,k}, \alpha_i, \beta_i)$. We define the matrix $\Gamma \in \mathbb{Z}^{n \times k}$ such that $\Gamma_{i,j} = \gamma_{i,j}$ and vectors $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top$ and $\vec{\beta} = (\beta_1, \ldots, \beta_n)^\top$.

2. Compute the matrix $M \in \mathbb{Z}^{(n-r)\times n}$ for Γ guaranteed by Lemma 6, where r is the rank of Γ over $\mathbb{Q}$, and compute $\vec{v} = M \cdot (-\vec{\alpha} - 2^T \cdot \vec{\beta}) \in \mathbb{Z}^{n-r}$.

3. For all $i \in [n-r]$ such that $\vec{v}_i \neq 0$, run the factoring algorithm of Lemma 2 with failure probability $2^{-2\lambda}$. If any succeed of the factoring attempts succeed and output p, q such that $N = p \cdot q$, output p, q. Otherwise, output $\perp$.

We start by analyzing the running time of A. In step 1, the algorithm samples $(32\lambda^4(k+1))/\epsilon^2$ potential vectors and determines the corresponding coefficients, which takes $\mathsf{poly}(\lambda, k, T) \cdot \mathsf{Time}_V(1^\lambda, N, \mathsf{st})$ time per choice of randomness. Step 2 can be computed in $\mathsf{poly}(\lambda, k, n, \log ||\Gamma||_{\max})$ time by Lemma 6. The factoring algorithm of Lemma 2 takes time $\mathsf{poly}(\lambda)$ to achieve the desired failure probability, which is repeated at most $n-k$ times. Thus, the entire algorithm A runs in expected $\mathsf{poly}(\lambda, k, T, 1/\epsilon(\lambda)) \cdot \mathsf{Time}_V(1^\lambda, N, \mathsf{st})$ time, as required.

We next show that A succeeds in outputting p, q such that $N = p \cdot q$ with probability at least $1 - 2^{-\lambda}$. First, by Claim 1, the probability that A aborts in Step 1 due to not finding n randomness values ρ_i such that $||\mathsf{coef}(\rho_i)||_{\max} < \frac{1}{2k} 2^{T/2(k+1)}$ is at most $2^{-2\lambda}$. It remains to analyze the probability A succeeds in factoring N in step 3.

Let $\vec{v} = M \cdot (-\vec{\alpha} - 2^T \cdot \vec{\beta}) \in \mathbb{Z}^n$ be the vector computed in step 2. In Claim 2, we show that, for all $x \in \mathbb{Z}_N^\star$ and each prime $\mathsf{prm} \in \{p, q\}$, there exists a solution $\vec{\pi} \in \mathbb{Z}^k$ such that $\Gamma \cdot \vec{\pi} = x|_{\mathsf{prm}} \cdot (-\vec{\alpha} - 2^T \cdot \vec{\beta}) \bmod (\mathsf{prm} - 1)$. This implies by Lemma 6 that $x|_{\mathsf{prm}} \cdot \vec{v} = \vec{0} \bmod (\mathsf{prm} - 1)$. Since this holds for every $x \in \mathbb{Z}_N^\star$, it must hold for x such that $x|_{\mathsf{prm}}$ is invertible mod $(\mathsf{prm} - 1)$, so we conclude that $\vec{v}$ must actually be equal to $\vec{0}$ mod $(\mathsf{prm} - 1)$. As this holds for both $\mathsf{prm} \in \{p, q\}$, it must be the case that $\vec{v} = \vec{0} \bmod \mathrm{lcm}(p-1, q-1) = \mathsf{Carm}(N)$. We consider two cases, either: (1) $\vec{v} = \vec{0}$ over $\mathbb{Z}$, or (2) there exists an i such that $\vec{v}_i = c \cdot \mathsf{Carm}(N)$ for some $c \in \mathbb{Z}$, $c \neq 0$. If case (2) holds, then the factoring algorithm of Lemma 2 will succeed with probability $1 - 2^{-2\lambda}$, and we are done.

We proceed to bound the probability that case (1) holds by $2^{-2\lambda}$. We note that by linearity, $M \cdot (-\vec{\alpha} - 2^T \vec{\beta}) = -M \cdot \vec{\alpha} - 2^T \cdot M \cdot \vec{\beta} = \vec{0}$. We consider two further subcases under case (1), either: (1A) $M \cdot \vec{\beta} = \vec{0}$, or (1B) there exists an index i such that $M_i \cdot \vec{\beta} \neq 0$. If case (1B) holds, this implies that

$$2^T \leq |M_i \cdot \vec{\beta}| \cdot 2^T \leq |M_i \cdot \vec{\alpha}|.$$

However, by Lemma 6, we know that

$$|M_i \cdot \vec{\alpha}| \leq (2k)^k \cdot ||\Gamma||_{\max}^{2k} \cdot ||\vec{\alpha}||_{\max}.$$

Since $||\Gamma||_{\max} < \frac{1}{2k} 2^{T/2(k+1)}$, it follows that

$$|M_i \vec{\alpha}| < (2k)^k \cdot \left(\frac{1}{2k} \cdot 2^{T/2(k+1)}\right)^{2k} \cdot \left(\frac{1}{2k} \cdot 2^{T/2(k+1)}\right) \leq 2^T,$$

in contradiction, so case (1B) cannot hold.

Moving on to case (1A) where $M \cdot \vec{\beta} = \vec{0}$ over $\mathbb{Z}$, note that this implies that $M \cdot \vec{\alpha} = \vec{0}$ over $\mathbb{Z}$ as well. Thus, for any $d \in \mathbb{Z}$ such that $x^d \neq x^{2^T}$, it holds that $M \cdot (-\vec{\alpha} - d \cdot \vec{\beta}) = \vec{0}$ over $\mathbb{Z}$. By Lemma 6, that implies that there exists $\vec{\pi} \in \mathbb{Z}^k$ such that $\Gamma \cdot \vec{\pi} = (-\vec{\alpha} - d \cdot \vec{\beta})$ over $\mathbb{Z}$, but that also implies the existence of a solution mod $(p-1)$ and $(q-1)$ and hence mod $\mathrm{lcm}(p-1, q-1) = \mathsf{Carm}(N)$ by Chinese Remainder theorem. But, by Lemma 5, the probability that this can be the case is at most $2^{-2\lambda}$.

Combining the above, we conclude that $A(1^\lambda, N, k, T, \mathsf{st}, 1/\epsilon)$ fails to factor N if A aborts at step 1, case (1) holds, or case (2) holds but the factoring algorithm fails. The probability of each of these three events is bounded by $2^{-2\lambda}$, so we conclude that A factors N with probability at least $1 - 3 \cdot 2^{-2\lambda} \geq 1 - 2^{-\lambda}$ as long as $\lambda \geq 2$, as required.

□

We proceed to provide proofs for Claims 1 and 2, and provide proofs of the lemmas for the analysis of case (1A) and (1B) in Sections 4.1 and 4.2, respectively.

For the following claim, recall that $\mathbb{Z}_N^\star \cong \mathbb{Z}_p^\star \times \mathbb{Z}_q^\star$, where each of the subgroups are cyclic. Let g_p and g_q be generators for the corresponding subgroups. Then, for any group element $h \in \mathbb{Z}_N^\star$, we denote integers $h|_p$ and $h|_q$ such that $h = g_p^{h|_p} \cdot g_q^{h|_q}$.

Claim 1. *A aborts during step 1 with probability at most $2^{-2\lambda}$.*

Proof. For each sampled $\rho \leftarrow \{0,1\}^\lambda$, the probability that $||\mathsf{coef}(\rho)||_{\max} < \frac{1}{2k} 2^{T/2(k+1)}$ is at least ϵ by assumption. Let $X_i = 1$ if the ith sampled ρ has small coefficients or 0 otherwise. Define $X = \sum_{i=1}^{2n\lambda/\epsilon^2} X_i$, and note that $\mathrm{E}[X] \geq 2n\lambda/\epsilon$. Then if A aborts, it must hold that $X < n$, which happens with probability at most

$$\Pr[|X - \mathrm{E}[X]| > 2n\lambda/\epsilon - n] \leq \Pr[|X - 2n\lambda/\epsilon| > n\lambda/\epsilon],$$

as $n\lambda/\epsilon \geq n$. By the Chernoff bound of Lemma 3, we can upper bound this quantity by

$$2e^{-2\left(\frac{n\lambda}{\epsilon}\right)^2 \cdot \frac{\epsilon^2}{2n\lambda}} \leq 2e^{-n\lambda} \leq 2^{-2\lambda},$$

where the last inequality holds since $n = 16\lambda^3(k+1) \geq 4$. □

Claim 2. *For all $x \in \mathbb{Z}_N^\star$ and each $\mathsf{prm} \in \{p, q\}$, there exists $\vec{\pi} \in \mathbb{Z}^k$ such that*

$$\Gamma \cdot \vec{\pi} = x|_{\mathsf{prm}} \cdot (-\vec{\alpha} - 2^T \cdot \vec{\beta}) \bmod (\mathsf{prm} - 1).$$

Proof. By completeness, for every $x \in \mathbb{Z}_N^\star$, there exists a k-element proof $z_1, \ldots, z_k$ that causes V to accept for every randomness $\rho \in \{0,1\}^\lambda$. In particular, for every $\rho \in \{0,1\}^\lambda$ with $\mathsf{coef}(\rho) = (\gamma_1, \ldots, \gamma_k, \alpha, \beta)^\top$, we have that

$$V(x, x^{2^T}, z_1, \ldots, z_k) = x^\alpha \cdot x^{2^T \cdot \beta} \cdot \prod_{i=1}^k z_i^{\gamma_i} = 1.$$

Fix some $\mathsf{prm} \in \{p, q\}$ with out loss of generality. We translate the fact that V accepts in $\mathbb{Z}_N^\star$ to a statement with respect to $\mathbb{Z}_{\mathsf{prm}}^\star$, so it must be the case that

$$g_p^{x|_{\mathsf{prm}} \cdot \alpha} \cdot g_p^{x|_{\mathsf{prm}} \cdot 2^T \cdot \beta} \cdot \prod_{i=1}^k g_p^{z_i|_{\mathsf{prm}} \cdot \gamma_i} = 1,$$

or equivalently, the exponents must satisfy

$$x|_{\mathsf{prm}} \cdot \alpha + x|_{\mathsf{prm}} \cdot 2^T \cdot \beta + \sum_{i=1}^k z_i|_{\mathsf{prm}} \cdot \gamma_i = 0 \bmod (\mathsf{prm} - 1)$$

$$\implies \sum_{i=1}^k z_i|_{\mathsf{prm}} \cdot \gamma_i = x|_{\mathsf{prm}} \cdot (-\alpha - 2^T \cdot \beta) \bmod (\mathsf{prm} - 1).$$

The same proof $z_1, \ldots, z_k$ must work for all randomness values ρ, so there must exist a solution $\vec{\pi} \in \mathbb{Z}^k$ given by $\vec{\pi}_i = z_i|_{\mathsf{prm}}$ such that $\Gamma \cdot \vec{\pi} = -\vec{\alpha} - 2^T \cdot \vec{\beta} \bmod (\mathsf{prm} - 1)$, as required. □

4.1 Case (1A) Analysis

We note that this section is not fully self-contained and relies on some notation from the proof of Theorem 3. In this section, we prove two lemmas that are used to show that if you sample enough vectors with small coefficients, by soundness, there cannot exist a solution $\vec{\pi}$ such that $\Gamma \cdot \vec{\pi} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \mathsf{Carm}(N)$. In Lemma 4, we show that each new vector must restrict the space of solutions with good probability, otherwise we can come up with a cheating prover that breaks soundness. In Lemma 5, we use this to show that if you sample a polynomial number of such vectors, it must restrict the space of solutions to the point that there can no longer exist a solution to the set of equations.

We recall the following notation defined in the preliminaries. Let p be a prime and $e \in \mathbb{N}$ an exponent. For a set of integer valued vectors V, we use the notation $\mathsf{Span}\,(p^e, V)$ to denote the set of all vectors spanned by V over the integers mod p^e.

Lemma 4. *Let $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$ such that $N = p \cdot q$ and $\mathsf{Carm}(N) = \prod_{j=1}^m p_j^{e_j}$ for unique primes p_j and $e_j \in \mathbb{N}$. Let $d \in \mathbb{Z}$ such that $x^d \neq y \in \mathbb{Z}_N^\star$. For any $\ell \geq 0$, let $\rho_1, \ldots, \rho_\ell \in \{0,1\}^\lambda$ with d-coefficient vectors $\mathsf{dcoef}(\rho_i, d) = (\gamma_{i,1}, \ldots, \gamma_{i,k}, \alpha_i + d\beta_i)^\top$ for $i \in [\ell]$ with corresponding coefficient matrix Γ and vectors $\vec{\alpha}$ and $\vec{\beta}$ such that there exists a vector $\vec{\pi}$ such that $\Gamma \cdot \vec{\pi} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \mathsf{Carm}(N)$. Then,*

$$\Pr_\rho \left[\forall j \in [m], \mathsf{dcoef}(\rho) \in \mathsf{Span}\left(p_j^{e_j}, \{\mathsf{dcoef}(\rho_i) : i \in [\ell]\}\right) \;\middle|\; ||\mathsf{coef}(\rho)||_{\max} < \frac{1}{2k} 2^{T/2(k+1)}\right] \leq 1/2.$$

Proof. Suppose the probability is greater than 1/2, then we construct a cheating prover $P^\star$ that convinces V that $f_{N,T}(x) = x^d \bmod N$ with fresh randomness ρ as follows.

$P^\star$ hardcodes $\rho_1, \ldots, \rho_\ell$ and computes a vector $\vec{\pi}$ such that $\Gamma \cdot \vec{\pi} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \varphi(N)$, which must exist by assumption. $P^\star$ outputs the proof elements $z_1, \ldots, z_k$ where $z_s = x^{\pi_s}$ for all $s \in [k]$.

Let $\rho \leftarrow \{0,1\}^\lambda$ be the randomness sampled by V with d-coefficient vector $\mathsf{dcoef}(\rho) = (\gamma_1, \ldots, \gamma_k, \alpha + d\beta)$. This implies that V accepts $y = x^d \neq x^{2^T}$ if

$$x^\alpha y^\beta \prod_{s=1}^k z_s^{\gamma_s} = x^{\alpha + d\beta + \sum_{s=1}^k \pi_s \gamma_s} = 1,$$

which holds if

$$\alpha + d \cdot \beta + \sum_{s=1}^k \pi_s \gamma_s = 0 \bmod \mathsf{Carm}(N).$$

For simplicity of notation, let E_j be the event that $\mathsf{dcoef}(\rho) \in \mathsf{Span}\left(p_j^{e_j}, \{\mathsf{dcoef}(\rho_i) : i \in [\ell]\}\right)$. Suppose E_j holds for all $j \in [m]$, and fix any particular prime power $p_j^{e_j}$. By definition of span, there exist integers $c_1, \ldots, c_\ell$ such that $\mathsf{dcoef}(\rho) = \sum_{i=1}^{\ell} c_i \cdot \mathsf{dcoef}(\rho_i)$. This implies that

$$\begin{aligned}\alpha + d \cdot \beta + \sum_{s=1}^{k} \pi_s \gamma_s &= \sum_{i=1}^{\ell} c_i \cdot \left(\alpha_i + d \cdot \beta_i + \sum_{s=1}^{k} \gamma_{i,s} \pi_s \right) \\ &= \sum_{i=1}^{\ell} (c_i \cdot 0) = 0 \bmod p_j^{e_j}\end{aligned}$$

As this holds for all prime power divisors of $\mathsf{Carm}(N)$, it also holds that

$$\alpha + d \cdot \beta + \sum_{s=1}^{k} \pi_s \gamma_s = 0 \bmod \mathsf{Carm}(N).$$

Thus, $P^\star$ succeeds with probability at least

$$\begin{aligned}&\Pr_\rho \left[\forall j \in [m], E_j\right] \\ &\geq \Pr_\rho \left[\forall j \in [m], E_j \;\middle|\; ||\mathsf{coef}(\rho)||_{\max} < \frac{1}{2k} 2^{T/2(k+1)} \right] \\ &\quad \cdot \Pr_\rho \left[||\mathsf{coef}(\rho)||_{\max} < \frac{1}{2k} 2^{T/2(k+1)} \right] \\ &> 1/2 \cdot 2\delta = \delta,\end{aligned}$$

in contradiction with the soundness of (P, V). □

Lemma 5. *Let $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$ such that $N = p \cdot q$ and $\mathsf{Carm}(N) = \prod_{j=1}^{m} p_j^{e_j}$ for unique primes p_j and $e_j \in \mathbb{N}$. Let $d \in \mathbb{Z}$ such that $x^d \neq y \in \mathbb{Z}_N^\star$. Let $n = 16\lambda^3(k+1)$ and $\rho_1, \ldots, \rho_n$ be a collection of random values in $\{0,1\}^\lambda$ conditioned on satisfying $||\mathsf{coef}(\rho_i)||_{\max} < \frac{1}{2k} 2^{T/2(k+1)}$ for all $i \in [n]$. For each $i \in [n]$, let $\mathsf{coef}(\rho_i) = (\gamma_{i,1}, \ldots, \gamma_{i,k}, \alpha_i, \beta_i)^\top$ with corresponding coefficient matrix Γ and vectors $\vec{\alpha}$ and $\vec{\beta}$. Then,*

$$\Pr\left[\exists \vec{\pi}, \Gamma \cdot \vec{\pi} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \mathsf{Carm}(N)\right] \leq 2^{-2\lambda}.$$

Proof. Consider the augmented matrix $\Gamma_{\mathsf{aug}} = (\Gamma | (-\vec{\alpha} - d\vec{\beta})) \in \mathbb{Z}^{n \times (k+1)}$. First, we observe that any solution $\vec{\pi}$ to the system of equations $\Gamma \cdot \vec{\pi} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \mathsf{Carm}(N)$ is also a solution mod any prime power $p_j^{e_j}$ that divides $\mathsf{Carm}(N)$. Our high level plan is to show that after sampling $n' \geq 4\lambda^2(k+1)$ vectors that non-trivially reduce the space of possible solutions, then there cannot be any solutions to the equation $\Gamma \cdot \vec{\pi} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \mathsf{Carm}(N)$. By Lemma 4, each newly sampled vector will reduce the space of solutions with probability at least 1/2 (assuming there space of solutions is non-empty). So, the only way there are remaining solutions is if after sampling $n = 16(k+1)\lambda^3$ vectors, we get fewer than $n' = 4(k+1)\lambda^2$ "good" vectors that reduce the space of solutions. However, we reduce this to a coin flipping problem to show this bad event happens with probability at most $2^{-2\lambda}$.

We start by showing that after at least $n' = 4(k+1)\lambda^2$ vectors that reduce the space of solutions, there must be no solutions remaining. Since $\mathsf{Carm}(N) \leq 2^{2\lambda}$, we note that there can be at most

$m \leq 2\lambda$ many prime powers $p_j^{e_j}$ that divide $\mathsf{Carm}(N)$, and each exponent $e_j \leq 2\lambda$ as well. We fix a particular prime power $p_j^{e_j}$ and look at the solutions over $\mathbb{Z}$ mod $p_j^{e_j}$. Let $B = (\vec{b}^{(1)}, \ldots, \vec{b}^{(k+1)})$ be a basis that spans all possible vectors in $\mathbb{Z}_{p_j^{e_j}}^{k+1}$. This means that all vectors can be written in the form $\sum_{i=1}^{k+1} c_i \cdot \vec{b}^{(i)}$ for integers $c_1, \ldots, c_{k+1}$.

We prove by induction on the dimension $i \in [k+1]$ that there can be at most $i \cdot e_j$ successive vectors that cut the space of solutions down mod $p_j^{e_j}$ with respect to the first i dimensions based on the basis vectors $\vec{b}^{(1)}, \ldots, \vec{b}^{(i)}$. First, for the base case of $i = 1$, all possible solutions are of the form $c \cdot b^{(1)}$ for some integer c. Note that if c is invertible mod $p_j^{e_j}$, then such a vector spans the entire space. So, we want to find the values of c that spans the fewest vectors. This corresponds to values of c such that $\gcd(c, p_j^{e_j}) = p_j^{e_j - 1}$, which only covers p many points. The next smallest way to cut the space down is with c such that $\gcd(c, p_j^{e_j}) = p_j^{e_j - 2}$. This continues until the only way to cut down the space is with invertible c mod $p_j^{e_j}$, so there are at most e_j successive vectors that reduce the space of solutions until none are left.

Now suppose that this holds for $i - 1 \geq 1$ dimensions. Then for i dimensions, we want to cover as few points as possible, so we extend all of the previous vectors so that they cover no points in dimension i spanned by $b^{(i)}$. To cover dimension i, we again need vectors with ith components corresponding to $c \cdot b^{(i)}$ where $\gcd(c, p_j^{e_j})$ is equal to $p_j^{e_j}$ then $p_j^{e_j - 1}$ and so on until the gcd is 1. This adds e_j more vectors, which suffices for the induction.

Thus, for each prime power $p_j^{e_j}$, after $(k+1) \cdot e_j \leq 2(k+1)\lambda$ many vectors are sampled, there will be no solutions mod $p_j^{e_j}$. However, each vector can reduce the space potentially via different prime powers, so that implies after $m \cdot 2(k+1)\lambda \leq 4(k+1)\lambda^2 = n'$ many "good" vectors, there will be no possible solutions left.

We now bound the corresponding coin-flipping problem. We sample $n = 16\lambda^3(k+1)$ many vectors, each which curs the space of solutions with probability at least 1/2. So we need to bound the probability that fewer than $n' = 4\lambda^2(k+1)$ of the n coin flips come up heads. To do so, let C_i denote a random variable which is 1 with probability 1/2 and 0 otherwise. Let $C = \sum_{i=1}^{16\lambda^3(k+1)} C_i$, so $\mathrm{E}[C] = 8\lambda^3(k+1)$. By the Chernoff-Hoeffding bound of Lemma 3, it holds that

$$\begin{aligned}\Pr[|C - \mathrm{E}[C]| > 8\lambda^3(k+1) - 4\lambda^2(k+1)] &\leq \Pr[|C - \mathrm{E}[C]| > 4\lambda^3(k+1)] \\ &\leq 2e^{-2\left(4\lambda^3(k+1)\right)^2/(16\lambda^3(k+1))} = 2e^{-2\lambda^3(k+1)} \leq 2^{-2\lambda},\end{aligned}$$

where the last inequality holds as long as $2\lambda^3(k+1) \geq 2\lambda + 1$, which is true for $k \geq 1$ and $\lambda \geq 2$. As discussed above, this also bounds the probability that there exists $\vec{\pi}$ satisfying $\Gamma \cdot \vec{\pi} = -\vec{\alpha} - d \cdot \vec{\beta}$ mod $\mathsf{Carm}(N)$, as required. □

4.2 Case (1B) Analysis

The following lemma is self-contained and is a key ingredient in the proof of Theorem 3. At a high level, we show how to reduce the existence of a solution $\vec{\pi}$ to a system of equations $\Gamma \cdot \vec{\pi} = \vec{\alpha}$ mod Φ for some composite number Φ, to computing a linear function $M \cdot \vec{\alpha}$. Additionally, M is efficient to compute, depends only on Γ, and $||M \cdot \vec{\alpha}||_{\max}$ cannot grow too much relative to $||\Gamma||_{\max}$ and $||\vec{\alpha}||_{\max}$. One interpretation of this lemma is generalizing the corresponding result over a field, where it suffices to simply compute the rank or determinant of a matrix, to the setting of the integers mod Φ.

Towards proving the lemma, we provide proofs for two additional claims at the end of this subsection. These culminate in Claim 4 which shows that if $\vec{x}$ is a solution to $H \cdot \vec{x} = \vec{0} \bmod \Phi$ for a square, lower triangular matrix H, then for any vector $\vec{\alpha}$, it must be the case that $\vec{\alpha}^\top \cdot \vec{x} \cdot \det(H) = 0 \bmod \Phi$.

Lemma 6. *Let $n, k \in \mathbb{N}$ such that $n \geq k+1$, and let $\Phi \in \mathbb{N}$ be any modulus. For every matrix $\Gamma \in \mathbb{Z}^{n \times k}$ with rank r over $\mathbb{Q}$, there exists a matrix $M \in \mathbb{Z}^{(n-r) \times n}$ such that the following holds for all $\vec{\alpha} \in \mathbb{Z}^n$:*

(A) If there exists $\vec{\pi} \in \mathbb{Z}^k$ such that $\Gamma \cdot \vec{\pi} = \vec{\alpha} \bmod \Phi$, then $M \cdot \vec{\alpha} = \vec{0} \bmod \Phi$.

(B) If $M \cdot \vec{\alpha} = \vec{0}$ over $\mathbb{Z}$, then there exists $\vec{\pi} \in \mathbb{Z}^k$ such that $\Gamma \cdot \vec{\pi} = \vec{\alpha}$ over $\mathbb{Z}$.

Furthermore, M can be computed in time $\mathsf{poly}(\lambda, k, n, \log ||\Gamma||_{\max})$, and $||M\vec{\alpha}||_{\max} \leq (2k)^k \cdot ||\Gamma||_{\max}^{2k} \cdot ||\vec{\alpha}||_{\max}$ for all $\vec{\alpha} \in \mathbb{Z}^n$.

Proof. First, let $H \in \mathbb{Z}^{n \times k}$ be the lower triangular Hermite normal form for the matrix Γ, as specified in Definition 1. For simplicity, we permute the rows of H such that the top r rows and first r columns form an $r \times r$ lower triangular matrix, with all remaining rows below. Note that the remaining $k - r$ columns of H will be all zeroes since Γ has rank r. We can think of applying this transformation first to Γ, which does not change the space of solution. This means that $H = \Gamma \cdot U$ for $U \in \mathbb{Z}^{k \times k}$, where $\det(U) \in \{1, -1\}$ and $||\Gamma||_{\max} \leq D$. For each $i \in [n]$, $j \in [k]$, we denote each entry $H_{i,j}$ of H by $h_{i,j}$. Since $\det(U) \in \{1, -1\}$, it follows that H and Γ span the same lattice, so there is an integer solution $\vec{\pi} \in \mathbb{Z}^k$ to $\Gamma \cdot \vec{\pi} = \vec{\alpha}$ if and only if there is an integer solution $\vec{\pi'} \in \mathbb{Z}^k$ to $H \cdot \vec{\pi'} = \vec{\alpha}$.

We start by describing how to construct the matrix M at a high level. Our approach is to augment the matrix H with a vector $\vec{\alpha}$ to get the matrix $H^{(0)} = (H|\vec{\alpha})$. We then apply a sequence of right multiplications $U^{(1)}, \ldots, U^{(r)}$ by unimodular matrices in order to zero out the first r rows of the last column in the augmented matrix. Let $H^{(i)} = H^{(0)} \cdot U^{(1)} \cdot \ldots \cdot U^{(i)}$ be the ith such intermediate matrix where the first i values of the last column are zeroed out. For the final matrix $H^{(r)}$, let $c = \prod_{i=1}^{r} H_{i,i}^{(r)}$ be the product of the diagonal entries of the first r columns. We show there is a matrix M such that, for each $j \in [n-r]$, $M_j \cdot \vec{\alpha} = c \cdot H_{j+r,k+1}^{(r)}$, so M is a linear function that on input $\vec{\alpha}$ outputs the multiple c times the (potentially) non-zero entries in the final column of $H^{(r)}$.

We construct the sequence of matrices $U^{(1)}, \ldots, U^{(r)}$ in order as follows. Recall that our goal is to define these matrices such that, for all $i \leq r$, the matrix

$$(H|\vec{\alpha}) \cdot U^{(1)} \cdot \ldots \cdot U^{(i)} \in \mathbb{Z}^{n \times (k+1)}$$

will be lower triangular in the first k columns and the $k+1$st column will have a 0 in the first i rows. We start with the construction of $U^{(1)}$. If $\vec{\alpha}_1$ is already equal to 0, we set $U^{(1)}$ to be the identity and continue. Otherwise, let $g_1 = \gcd(h_{1,1}, \vec{\alpha}_1)$, and define $\mathsf{bz}_{1,L}, \mathsf{bz}_{1,R}$ to be Bezout coefficients such that $\mathsf{bz}_{1,L} \cdot h_{1,1} - \mathsf{bz}_{1,R} \cdot \vec{\alpha}_1 = g_1$. Then, we set

$$U^{(1)} = \begin{bmatrix} \mathsf{bz}_{1,L} & 0 & \cdots & 0 & -\vec{\alpha}_1/g_1 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ \mathsf{bz}_{1,R} & 0 & \cdots & 0 & h_{1,1}/g_1 \end{bmatrix} \in \mathbb{Z}^{k\times k}.$$

Note that by definition of g_1, $\mathsf{bz}_{1,L}$, $\mathsf{bz}_{1,R}$, $\det(U^{(1)}) = 1$ and $(-\vec{\alpha}_1/g_1), (h_{1,1}/g_1) \in \mathbb{Z}$. The resulting matrix has the form

$$H^{(1)} = H^{(0)} \cdot U^{(1)} = \begin{bmatrix} g_1 & 0 & 0 & \cdots & 0 \\ h'_{2,1} & h_{2,2} & 0 & \cdots & (h_{1,1}\vec{\alpha}_2 - h_{2,1}\vec{\alpha}_1)/g_1 \\ h'_{3,1} & h_{3,2} & h_{3,3} & \cdots & (h_{1,1}\vec{\alpha}_3 - h_{3,1}\vec{\alpha}_1)/g_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h'_{n,1} & h_{n,2} & h_{n,3} & \cdots & (h_{1,1}\vec{\alpha}_n - h_{n,1}\vec{\alpha}_1)/g_1 \end{bmatrix},$$

where $h'_{i,1} \in \mathbb{Z}$ for all $i \le n$. Let $\vec{\alpha}^{(1)} \in \mathbb{Z}^n$ be the last column of $H^{(1)}$.

We next define the matrix $U^{(2)}$. Again, if $\alpha^{(1)}_2 = 0$, $U^{(2)}$ will simply be the identity. Otherwise, we let $g_2 = \gcd(h_{2,2}, \vec{\alpha}^{(1)}_2)$ and let $\mathsf{bz}_{2,L}$, $\mathsf{bz}_{2,R}$ be Bezout coefficients such that $\mathsf{bz}_{2,L} \cdot h_{2,2} - \mathsf{bz}_{2,R} \cdot \vec{\alpha}^{(1)}_2 = g_2$. Then, we set

$$U^{(2)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \mathsf{bz}_{2,L} & 0 & \cdots & -\vec{\alpha}^{(1)}_2/g_2 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \mathsf{bz}_{2,R} & 0 & \cdots & h_{2,2}/g_2 \end{bmatrix} \in \mathbb{Z}^{k\times k}.$$

We continue this process to compute $U^{(3)}, \ldots, U^{(r)}$ and eventually get the matrix

$$H^{(r)} = H^{(0)} \cdot U^{(1)} \cdot \ldots \cdot U^{(r)} = \begin{bmatrix} g_1 & 0 & \cdots & 0 & \cdots & 0 \\ h'_{2,1} & g_2 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ h'_{r,1} & h'_{r,2} & \cdots & g_r & \cdots & 0 \\ h'_{r+1,1} & h'_{r+1,2} & \cdots & h'_{r+1,r} & \cdots & \alpha^{(r)}_{r+1} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ h'_{n,1} & h'_{n,2} & \cdots & h'_{n,r} & \cdots & \vec{\alpha}^{(r)}_n \end{bmatrix}.$$

We define the vector $\vec{\beta}$ such that, for all $i \in [n-r]$, $\vec{\beta}_i = \vec{\alpha}^{(r)}_{i+r} \cdot \prod_{s=1}^{r} g_s$. We claim that the values $\vec{\beta}$ are of the form $M \cdot \alpha$ for a matrix M which is defined based on the entries of H. Recall that $\vec{\alpha}^{(i)} \in \mathbb{Z}^n$ is the last column of the matrix $H^{(i)}$, and similarly let $\vec{\alpha}^{(0)} = \vec{\alpha}$. We can write a recursive formula for the value of each entry in $\vec{\alpha}^{(i)}$ based on $\vec{\alpha}^{(i-1)}$ using $U^{(i-1)}$. Namely, for all $i \le r$ and $i \le j \le n$, $\alpha^{(i)}_j$ is given by

$$\vec{\alpha}^{(i)}_j = \frac{1}{g_i} \cdot \left(h_{i,i} \cdot \vec{\alpha}^{(i-1)}_j - h_{j,i} \cdot \vec{\alpha}^{(i-1)}_i \right).$$

Using this recursive expression, we prove the following subclaims by induction on $i = 0, 1, \dots, r$ below:

(1) For all $i \le j \le n$, $\prod_{s=1}^{i} g_s \cdot \vec{\alpha}_j^{(i)}$ is a linear function of $\vec{\alpha}$.

(2) For all $i \le j \le n$, there exist 2^i sets $S_1, \dots, S_{2^i}$ such that each set contains $i-1$ elements, from distinct rows in H from the set of rows $\{1, \dots, i-1, j\}$. It holds that $\left|\prod_{s=1}^{i} g_s \cdot \vec{\alpha}_j^{(i)}\right| \le ||\vec{\alpha}||_{\max} \cdot \sum_{\ell=1}^{2^i} \cdot \prod_{a \in S_\ell} a$.

By setting $i = r$ in (1), we conclude that there exists a matrix $M \in \mathbb{Z}^{(n-r)\times k}$ such that $\vec{\beta} = M \cdot \vec{\alpha}$. Setting $i = r$ for (2), we notice that for each set S_ℓ, if it contains an element from row $j \le r$, it is upper bounded by $h_{j,j}$ by the row reduced property of HNF. Thus, the product of these elements is at most $\prod_{j=1}^{r} h_{j,j}$, which is an upper bound on the determinant of the matrix. This is bounded by $||\Gamma||_{\max}^{k} \cdot k^{k/2}$ by Hadamard's inequality. The potential element from the row $j > r$ is simply at most $||H||_{\max} \le ||\Gamma||^k$. Thus, we conclude that $\vec{\beta} \le (2k)^k \cdot ||\Gamma||_{\max}^{2k} \cdot ||\vec{\alpha}||_{\max}$, as required.

Now to prove (1), note that for $i = 0$ and $0 \le j \le n$, $\prod_{s=1}^{0} g_s \cdot \vec{\alpha}_j^{(0)} = \vec{\alpha}_j$, so this is clearly a linear function in $\vec{\alpha}$. Assume inductively that the claim holds for $i - 1 \ge 0$. For any $i \le j \le n$, we can write

$$\prod_{s=1}^{i} g_s \cdot \vec{\alpha}_j^{(i)} = h_{i,i} \cdot \prod_{s=1}^{i-1} g_s \cdot \vec{\alpha}_j^{(i-1)} - h_{j,i} \cdot \prod_{s=1}^{i-1} g_s \cdot \vec{\alpha}_i^{(i-1)}. \tag{$\star$}$$

However, the individual terms $\prod_{s=1}^{i-1} g_s \cdot \vec{\alpha}_j^{(i-1)}$ and $\prod_{s=1}^{i-1} g_s \cdot \vec{\alpha}_{i-1}^{(i-1)}$ are linear in $\vec{\alpha}$ by assumption, so the combined term $\prod_{s=1}^{i} g_s \cdot \vec{\alpha}_j^{(i)}$ is also linear in $\vec{\alpha}$, as required.

We next prove (2). For the base case of $i = 0$, note that for all $1 \le j \le n$, we can take the set $S_{2^0} = \emptyset$ and note that the empty product is simply 1. It follows that $\vec{\alpha}_j^{(0)} = 1 \cdot \vec{\alpha}_j \le ||\vec{\alpha}||_{\max}$, as required for the base case. Assume inductively this holds for $i - 1 \ge 0$. We refer to the individual terms in Equation ($\star$). Let $S_1, \dots, S_{2^{i-1}}$ be the corresponding sets for the first term in Equation ($\star$) and $S'_1, \dots, S'_{2^{i-1}}$ be the sets for the second term. Then,

$$\left| h_{i,i} \cdot \prod_{s=1}^{i-1} g_s \cdot \vec{\alpha}_j^{(i-1)} \right| \le h_{i,i} \cdot \left(||\vec{\alpha}||_{\max} \cdot \sum_{\ell=1}^{2^{i-1}} \cdot \prod_{a \in S_\ell} a \right),$$

and for the second term

$$\left| h_{j,i} \cdot \prod_{s=1}^{i-1} g_s \cdot \vec{\alpha}_i^{(i-1)} \right| \le h_{j,i} \cdot \left(||\vec{\alpha}||_{\max} \cdot \sum_{\ell=1}^{2^{i-1}} \cdot \prod_{a \in S'_\ell} a \right).$$

Combining these terms, note that for all $\ell \in [2^{i-1}]$, S_ℓ contains only elements from row j and rows in $[i-1]$, so if we add $h_{i,i}$ to each of these sets, they now contain $i-1$ unique elements from row j and $[i]$. For S'_ℓ, these elements contain elements only from $[i]$, so if we add $h_{j,i}$ to each of these sets, they also now contain $i-1$ unique elements from j and $[i]$. This results in 2^i new sets $S_1^\star, \dots, S_{2^i}^\star$ such that

$$\left| \prod_{s=1}^{i} g_s \cdot \vec{\alpha}_j^{(i)} \right| \le ||\vec{\alpha}||_{\max} \cdot \sum_{\ell=1}^{2^i} \cdot \prod_{a \in S_\ell^\star} a,$$

as required.

To conclude the proof of the lemma, we show why the matrix M such that $M \cdot \vec{\alpha} = \vec{\beta}$ satisfies (A) and (B) from the lemma statement. For (A), suppose there exists $\vec{\pi}$ such that $\Gamma \cdot \vec{\pi} = \vec{\alpha} \bmod \Phi$. As $H^{(r)}$ is constructed by applying right unimodular multiplications to Γ, then the first r columns of $H^{(r)}$ span the same space as Γ. So, if we let S be the square, lower triangular matrix from the first r rows and columns of $H^{(r)}$, then we know that there exists a vector $\vec{x}$ such that $S \cdot \vec{x} = \vec{0} \bmod \Phi$ and $H_j^{(r)} \cdot \vec{x} = \vec{\alpha}_j^{(r)}$ for all $j \in [r+1, n]$. However, by Claim 4, we have that $\vec{\alpha}_j^{(r)} \cdot \prod_{s=1}^{r} g_s = \vec{\beta}_{j-r} = \vec{0} \bmod \Phi$. So $M \cdot \vec{\alpha} = \vec{\beta} = \vec{0} \bmod \Phi$, as required.

For (B), suppose now that $M \cdot \vec{\alpha} = \vec{0}$ over $\mathbb{Z}$. This implies that for each $j \in [n-r]$, $\vec{\beta}_j = \vec{\alpha}_{j+r}^{(r)} \cdot \prod_{s=1}^{r} g_s = 0$. As $g_s \neq 0$ for all $s \in [r]$, this means that $\vec{\alpha}_{j+r}^{(r)} = 0$ for all $j \in [n-r]$, so $\vec{\alpha}^{(r)} = \vec{0} \in \mathbb{Z}^n$. This implies that $\vec{0} \in \mathbb{Z}^k$ is a solution to the system of equations specified by $H^{(r)}$ over $\mathbb{Z}$, so there must also exist an integer solution $\vec{\pi}$ such that $\Gamma \cdot \vec{\pi} = \vec{\alpha}$ over $\mathbb{Z}$. $\square$

Claim 3. *Let $\Phi \in \mathbb{N}$. Let $H \in \mathbb{Z}^{k \times k}$ be a lower triangular, square matrix. Let $\vec{x} \in \mathbb{Z}^k$ be a vector such that $H \cdot \vec{x} = \vec{0} \bmod \Phi$. Then for all $i \in [k]$,*

$$x_i \cdot \prod_{s=1}^{i} h_{s,s} = 0 \bmod \Phi.$$

Proof. This is true for $i = 1$ as $h_{1,1} \cdot x_1 = 0 \bmod \Phi$ since $H \cdot \vec{x} = \vec{0} \bmod \Phi$. Let $i > 1$ and assume it is true for all $j < i$ inductively. By the following sequence of equalities, we conclude that

$$\begin{aligned} x_i \cdot \prod_{s=1}^{i} h_{s,s} &= \sum_{j=1}^{i-1} (-1 \cdot h_{i,j} \cdot x_j) \cdot \prod_{s=1}^{i-1} h_{s,s} \\ &= \sum_{j=1}^{i-1} \left(x_j \cdot \prod_{s=1}^{j} h_{s,s} \right) \cdot \left(-1 \cdot h_{i,j} \cdot \prod_{s=j+1}^{i-1} h_{s,s} \right) \\ &= 0 \bmod \Phi \end{aligned}$$

$\square$

Claim 4. *Let $H \in \mathbb{Z}^{k \times k}$ be a lower triangular, square matrix. Let $\vec{x} \in \mathbb{Z}^k$ be a vector such that $H \cdot \vec{x} = \vec{0} \bmod \Phi$. For any vector $\vec{\alpha} \in \mathbb{Z}^n$, if $\vec{\alpha}^\top \cdot \vec{x} = \gamma \bmod \Phi$, then*

$$\gamma \cdot \prod_{s=1}^{k} h_{s,s} = 0 \bmod \Phi.$$

Proof. This follows from the previous claim via the following sequence of equalities:

$$\begin{aligned} \gamma \cdot \prod_{s=1}^{k} h_{s,s} &= \left(\sum_{i=1}^{k} \alpha_i x_i \right) \cdot \prod_{s=1}^{k} h_{s,s} \\ &= \sum_{i=1}^{k} \left(x_i \cdot \prod_{s=1}^{i} h_{s,s} \right) \cdot \left(\alpha_i \cdot \prod_{s=i+1}^{k} h_{s,s} \right) \\ &= 0 \bmod \Phi. \end{aligned}$$

$\square$

4.3 Extension to General Hidden Order Groups

Let $\mathbb{G}$ be any finite, abelian, multiplicative group. For any $\lambda \in \mathbb{N}$, we let $\mathsf{GroupGen}(1^\lambda)$ be an algorithm that outputs some group of size $[2^\lambda, 2^\lambda + 1)$ such that it is believe that it is hard to compute the order of a random group $\mathbb{G} \leftarrow \mathsf{GroupGen}(1^\lambda)$. Any such group $\mathbb{G}$ must be finitely generated, so there exist elements $g_1, \ldots, g_s$ such that every $h \in \mathbb{Z}_N^\star$ is equal to $\prod_{i=1}^{s} g_i^{h|_i}$, where $h|_i \in \mathbb{Z}$ is the ith component of h. We use $\mathsf{ord}(\mathbb{G})$ to denote the size of the group, and $\mathsf{ord}(g)$ to denote the minimum c such that $g^c = 1$. Borrowing notation from $\mathbb{Z}_N^\star$, we use $\mathsf{Carm}(\mathbb{G})$ to denote the maximum value of $\mathsf{ord}(g)$ for any $g \in \mathbb{G}$. In particular, there must exist some $g \in \mathbb{G}$ such that $\mathsf{ord}(g) = \mathsf{Carm}(\mathbb{G})$.

The proof of Theorem 3 goes through by considering an arbitrary group $\mathbb{G}$ via the following modifications. We consider inputs x such that $\mathsf{ord}(x) = \mathsf{Carm}(\mathbb{G})$, so we can equivalently work in the exponent mod $\mathsf{Carm}(\mathbb{G})$. Claim 2 easily generalizes to the setting of any set of $s \geq 1$ generators for $\mathbb{G}$, which implies a solution mod $\mathsf{ord}(g_i)$ for all $i \in [s]$ and hence mod $\operatorname{lcm}(\{\mathsf{ord}(g_i) : i \in [s]\}) = \mathsf{Carm}(\mathbb{G})$. Lemmas 4 and 5 only depend on the prime power factorization of $\mathsf{Carm}(\mathbb{G})$ and not any specific structure of $\mathbb{Z}_N^\star$. Lemma 6 is generic for any system of equations mod Φ, so that still holds mod $\mathsf{Carm}(\mathbb{G})$. Finally, case (2) shows that we can compute a multiple of $\mathsf{Carm}(\mathbb{G})$, which just happens that in the particular case of $\mathbb{Z}_N^\star$, implies a factoring algorithm. Therefore, we immediately get the following corollary.

Corollary 1. *Let $\lambda \geq 2, T \in \mathbb{N}$, $k \colon \mathbb{N} \to \mathbb{N}$, $\delta, \epsilon \colon \mathbb{N} \to [0,1]$, $\mathbb{G} \in \operatorname{Supp}\left(\mathsf{GroupGen}(1^\lambda)\right)$, and (P, V) be a k-element straight-line generic-group proof system for the function $f_{\mathbb{G},T}(x) = x^{2^T} \in \mathbb{G}$ with soundness error δ. For any $(\mathsf{st}, \pi_1, \ldots, \pi_{k(\lambda)}) \in \operatorname{Supp}\left(P(1^\lambda, \mathbb{G}, T, x, f_{\mathbb{G},T}(x))\right)$. If*

$$\Pr_{\rho}\left[\mathsf{ParTime}_V(1^\lambda, \mathbb{G}, T, \mathsf{st}) < \frac{T}{2(k(\lambda)+1)} - \log(2k(\lambda))\right] \geq \max(2\delta(\lambda), \epsilon(\lambda)),$$

then there exists a standard model probabilistic $\mathsf{poly}(\lambda, k(\lambda), T, 1/\epsilon(\lambda)) \cdot \mathsf{Time}_V(1^\lambda, \mathbb{G}, \mathsf{st})$ time algorithm A such that

$$\Pr\left[c \leftarrow A\left(1^\lambda, \mathbb{G}, k, T, \mathsf{st}, 1/\epsilon(\lambda)\right) : \mathsf{ord}(\mathbb{G}) \text{ divides } c\right] \geq 1 - 2^{-\lambda}.$$

5 Recursive Interactive Proofs

Recall that we consider proofs for functions of the form $f_{N,T}(x) = x^{2^T} \bmod N$ for $\lambda \in \mathbb{N}$ and $N \in \operatorname{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$. We show how our one round lower bound implies a lower bound for "recursive" interactive proofs. At a high level, these are public-coin proofs where after each round i, P and V compute group elements x_i, y_i and a time bound T_i. Then, they start a new proof (recursively) for the claim that $f_{N,T_i}(x_i) = x^{2^{T_i}} \bmod N$. We use $\langle P, V\rangle_{\mathsf{rec}}(1^\lambda, N, x, y, T, r, k)$, where r is the number of rounds and k is the bound on the number of group elements sent by P in each round, to denote the output of V in the following interaction:

1. P sends a bit string $\mathsf{st} \in \{0,1\}^*$ and k group elements $\pi_1, \ldots, \pi_k$ to V.
2. V responds with a random string $\rho \leftarrow \{0,1\}^\lambda$.
3. If $r > 1$, P and V both compute $(x', y', T') = A(1^\lambda, N, x, y, \mathsf{msg}, \rho)$ for a generic group algorithm A. P and V then execute an $r-1$ round recursive proof for the statement $y' = f_{N,T'}(x') = (x')^{2^{T'}} \bmod N$.

4. If $r = 1$, V outputs $g = A(1^\lambda, N, \mathsf{msg}; \rho)$ for a generic algorithm A.

In the above protocol, let $A_1, \dots, A_r$ be the generic algorithms in each round. We define $\mathsf{Time}_V = \sum_{i=1}^{r} \mathsf{Time}_{A_i}$, and similarly for $\mathsf{ParTime}_V$ and Width_V. Given the syntax above, we define a recursive proof system as follows.

Definition 3 (Recursive Generic Group Proofs)**.** Let $T \in \mathbb{N}$, $\delta \colon \mathbb{N} \to [0,1]$, and $k, r \colon \mathbb{N} \to \mathbb{N}$. For any $\lambda \in \mathbb{N}$, $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$, let $f_{N,T} \colon \mathbb{Z}_N^\star \to \mathbb{Z}_N^\star$ be the function such that $f_{N,T}(x) = x^{2^T} \bmod N$. We say that the pair (P, V) is a k-element, r-round *recursive* generic group proof system for $f_{N,T}$ with δ soundness if for all $\lambda \in \mathbb{N}$, $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$, $k = k(\lambda)$, and $r = r(\lambda)$, the following holds, where (x_i, y_i, T_i) is the input statement at round $i \in [r]$:

- Completeness: For all $x \in \mathbb{Z}_N^\star$, it holds that

$$1 = \langle P, V \rangle_{\mathsf{rec}}(1^\lambda, N, x, f_{N,T}(x), T, r, k).$$

  Furthermore, $f_{N,T_i}(x_i) = y_i$ for all $i \in [r]$.

- Soundness: For all $x \in \mathbb{Z}_N^\star$, $y \neq f_{N,T}(x)$, and algorithms $P^\star$

$$\Pr\left[1 = \langle P^\star, V \rangle_{\mathsf{rec}}(1^\lambda, N, x, y, T, r, k)\right] \leq \delta(\lambda).$$

  Furthermore, if $f_{N,T_i}(x_i) \neq y_i$ for a round $i \in [r-1]$, then $\Pr[f_{N,T_{i+1}}(x_{i+1}) = y_{i+1}] \leq \delta(\lambda)$.

We are now ready to state our main result for recursive interactive proofs.

Theorem 4. *Let $\lambda \geq 2, T \in \mathbb{N}$, $k, r \colon \mathbb{N} \to \mathbb{N}$, $\delta, \epsilon \colon \mathbb{N} \to [0,1]$, $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$, and (P, V) be a k-element, r-round recursive straight-line generic group proof system for $f_{N,T}$ with soundness error δ. Let $\vec{\mathsf{st}} \in (\{0,1\}^*)^r$ be any sequence of bit strings where $\vec{\mathsf{st}}_i$ is an explicit bit string in the support of P for round i. If*

$$\Pr\left[\mathsf{ParTime}_V(1^\lambda, N) < \frac{T}{3 \cdot (2(k(\lambda)+1))^{r(\lambda)}} - 6 \cdot \log(2k(\lambda))\right] \geq \max(2\delta(\lambda), \epsilon(\lambda)),$$

then there exists a standard model probabilistic $\mathsf{poly}(\lambda, k(\lambda), T, r(\lambda), 1/\epsilon(\lambda)) \cdot \mathsf{Time}_V(1^\lambda, N, \vec{\mathsf{st}})$ time algorithm A such that

$$\Pr\left[p, q \leftarrow A\left(1^\lambda, N, k(\lambda), T, r(\lambda), \vec{\mathsf{st}}, 1/\epsilon(\lambda)\right) : N = p \cdot q\right] \geq 1 - 2^{-\lambda}.$$

Proof. For each round $i \in [r]$, let (x_i, y_i, T_i) be the statement of the round, so P is trying to convince V that $y_i = x_i^{2^{T_i}} \bmod N$. In particular, (x_1, y_1, T_1) is simply the input statement (x, y, T). Let A_i be the generic group algorithms used in each round to compute the next statement $(x_{i+1}, y_{i+1}, T_{i+1})$. We emphasize that (x_i, y_i, T_i) are all random variables for $i \geq 2$.

We define the following two events: $\mathsf{E1}$ and $\mathsf{E2}$. Let $\mathsf{E1}$ be the event that $\mathsf{ParTime}_V(1^\lambda, N) < T/(3 \cdot (2(k+1))^r) - 6 \cdot \log(2k)$. Let $\mathsf{E2}$ be the event where there exists an $i \in [r]$ such that $\mathsf{ParTime}_{A_i}(1^\lambda, N, \mathsf{st}_i) + T_{i+1} < T_i/2(k+1) - \log(2k)$, where we define $T_{r+1} = 0$. Below, we first show (1) that $\neg\mathsf{E2}$ implies $\neg\mathsf{E1}$, and hence $\mathsf{E1}$ implies $\mathsf{E2}$. Then, we show (2) that if $\Pr[\mathsf{E2}] \geq \max(2\delta, \epsilon)$, this implies a factoring algorithm by Theorem 3. (1) and (2) suffice to prove the theorem since together they show that if $\Pr[\mathsf{E1}] \geq \max(2\delta, \epsilon)$, then there exists a factoring algorithm, and $\mathsf{E1}$ is the event assumed in the theorem statement, as required.

For (1), we assume that $\neg\mathsf{E2}$ holds and want to show that implies $\neg\mathsf{E1}$ must hold. This means that for all $i \in [r]$, it holds that $\mathsf{ParTime}_{A_i} + T_{i+1} \geq T_i/2(k+1) - \log(2k)$, where $T_{r+1} = 0$. We use the fact that $\mathsf{ParTime}_V \geq \mathsf{ParTime}_{A_i}$ for all $i \in [r]$, and we also rearrange the above to conclude that $T_{i+1} > T_i/2(k+1) - \log(2k) - \mathsf{ParTime}_{A_i}$. We then get a lower bound on $\mathsf{ParTime}_{A_r}$ via the following sequence of inequalities.

$$\begin{aligned}
\mathsf{ParTime}_{A_r} &\geq \frac{T_r}{2(k+1)} - \log(2k) \\
&\geq \frac{T_{r-1}/2(k+1) - \log(2k) - \mathsf{ParTime}_{A_{r-1}}}{2(k+1)} - \log(2k) \\
&\geq \frac{T_{r-1}}{(2(k+1))^2} - (\log(2k) - \mathsf{ParTime}_V) \cdot \left(1 + \frac{1}{2(k+1)}\right) \\
&\geq \frac{T_{r-2}}{(2(k+1))^3} - (\log(2k) - \mathsf{ParTime}_V) \cdot \left(1 + \frac{1}{2(k+1)} + \frac{1}{(2(k+1))^2}\right) \\
&\cdots \\
&\geq \frac{T_1}{(2(k+1))^r} - (\log(2k) - \mathsf{ParTime}_V) \cdot \sum_{i=0}^{r-1} \frac{1}{(2(k+1))^i} \\
&\geq \frac{T}{(2(k+1))^r} - 2 \cdot (\log(2k) - \mathsf{ParTime}_V),
\end{aligned}$$

where the last line holds since $1/(2(k+1)) \geq 1/2$. Using that $\mathsf{ParTime}_{A_r} \leq \mathsf{ParTime}_V$, we rearrange terms to get that

$$\mathsf{ParTime}_V \geq \frac{T}{3(2(k+1))^r} - 6\log(2k),$$

which is the event $\neg\mathsf{E1}$, as desired.

For (2), recall that we want to show if $\Pr[\mathsf{E2}] \geq \max(2\delta, \epsilon)$, then we can construct a factoring algorithm as stated in the theorem. As $\mathsf{E2}$ holds, this implies that there exists an $i \in [r]$ such that $\mathsf{ParTime}_{A_i} + T_{i+1} < T_i/2(k+1) - \log(2k)$. We focus on the case where $i < r$. The case of $i = r$ follows similarly however there the verifier does not need to perform any additional squarings at the end.

We construct a proof system $(\widehat{P}, \widehat{V})$ for the repeated squaring with exponent 2^{T_i} as follows. $\widehat{P}$ simply outputs whatever P would have output in round i of the recursive IP on input (x, y, T_i), call it $(\mathsf{st}, \pi_1, \ldots, \pi_k)$. $\widehat{V}$ then samples random coins $\rho \leftarrow \{0,1\}^\lambda$ and runs $A_i(1^\lambda, N, T_i, x, y, \mathsf{st}, \pi_1, \ldots, \pi_k, \rho)$ computing group elements x', y' and a time bound T_{i+1}. $\widehat{V}$ outputs $(x')^{2^{T_{i+1}}} \cdot (y')^{-1}$. It is clear that $\widehat{V}$ runs in parallel time $\mathsf{ParTime}_{A_i} + T_{i+1} < T_i/2(k+1) - \log(2k)$ with probability at least $\max(2\delta, \epsilon)$, by assumption. To apply Theorem 3, it remains to show that completeness and soundness hold.

For completeness of $(\widehat{P}, \widehat{V})$, note that completeness of the recursive proof guarantees that $(x')^{2^{T_{i+1}}} = y'$. Since $\widehat{V}$ checks this directly, it follows that $\widehat{V}$ outputs 1 as required. For soundness, we rely on the fact that if $(x)^{2^{T_i}} \neq y$, then it must be the case that $(x')^{2^{T_{i+1}}} \neq y'$ with probability at least $1 - \delta$. Therefore, $\widehat{V}$ will accept with probability at most δ, as required.

In order to construct the factoring algorithm A, we run the algorithm A' guaranteed by Theorem 3 given $(\widehat{P}, \widehat{V})$ on input $(1^\lambda, N, k, T, \vec{\mathsf{st}}_i, 1/\epsilon)$ for all $i \in [r]$. This increases the running time by a factor of r, but will still succeed with probability $1 - 2^{-\lambda}$ when run on round i, achieving the desired success probability. □

Acknowledgements. Cody Freitag's work was done during an internship at NTT Research. He is also supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. DGE–2139899 and DARPA Award HR00110C0086. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Defense Advanced Research Projects Agency (DARPA).

Ilan Komargodski was supported in part by an Alon Young Faculty Fellowship, by a JPM Faculty Research Award, by a grant from the Israel Science Foundation (ISF Grant No. 1774/20), and by a grant from the US-Israel Binational Science Foundation and the US National Science Foundation (BSF-NSF Grant No. 2020643).

References

[AM16] Divesh Aggarwal and Ueli Maurer. Breaking RSA generically is equivalent to factoring. *IEEE Trans. Inf. Theory*, 62(11):6251–6259, 2016.

[Bab85] László Babai. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, STOC*, pages 421–429, 1985.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 106–115, 2001.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Advances in Cryptology - CRYPTO*, pages 757–788, 2018.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from DARK compilers. In *Advances in Cryptology - EUROCRYPT*, pages 677–706, 2020.

[BHR+21] Alexander R. Block, Justin Holmgren, Alon Rosen, Ron D. Rothblum, and Pratik Soni. Time- and space-efficient arguments from groups of unknown order. In *Advances in Cryptology - CRYPTO*, pages 123–152, 2021.

[BL18] Nir Bitansky and Huijia Lin. One-message zero knowledge and non-malleable commitments. In *Theory of Cryptography - 16th International Conference, TCC*, pages 209–234, 2018.

[BN00] Dan Boneh and Moni Naor. Timed commitments. In *Advances in Cryptology - CRYPTO*, pages 236–254, 2000.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1082–1090, 2019.

[DKP21] Dana Dachman-Soled, Ilan Komargodski, and Rafael Pass. Non-malleable codes for bounded parallel-time tampering. In *Advances in Cryptology - CRYPTO*, pages 535–565, 2021.

[EFKP20a] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. In *Advances in Cryptology - EUROCRYPT*, pages 125–154, 2020.

[EFKP20b] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. SPARKs: Succinct parallelizable arguments of knowledge. In *Advances in Cryptology - EUROCRYPT*, pages 707–737, 2020.

[FJK21] Rex Fernando, Aayush Jain, and Ilan Komargodski. Maliciously-secure MrNISC in the plain model. *IACR Cryptol. ePrint Arch.*, page 1319, 2021.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In *Advances in Cryptology - CRYPTO*, pages 33–62, 2018.

[FKPS21] Cody Freitag, Ilan Komargodski, Rafael Pass, and Naomi Sirkin. Non-malleable time-lock puzzles and applications. In *Theory of Cryptography - 19th International Conference, TCC*, pages 447–479, 2021.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO*, pages 186–194, 1986.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, 1998.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *44th Symposium on Foundations of Computer Science (FOCS 2003)*, pages 102–113, 2003.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27:1–27:64, 2015.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GVW02] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Comput. Complex.*, 11(1-2):1–53, 2002.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[JS13] Tibor Jager and Jörg Schwenk. On the analysis of cryptographic assumptions in the generic ring model. *J. Cryptol.*, 26(2):225–245, 2013.

[KB14] Swastik Kopparty and Abhishek Bhrushundi. Lecture 3: Finding integer solutions to systems of linear equations, Fall 2014.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, STOC*, pages 723–732, 1992.

[KLX20] Jonathan Katz, Julian Loss, and Jiayu Xu. On the security of time-lock puzzles and timed commitments. In *Theory of Cryptography – TCC*, pages 390–413, 2020.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[LPS20] Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. *SIAM J. Comput.*, 49(4), 2020.

[LV20] Alex Lombardi and Vinod Vaikuntanathan. Fiat-shamir for repeated squaring with applications to ppad-hardness and vdfs. In *Advances in Cryptology - CRYPTO*, pages 632–651, 2020.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography. In *IMACC*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th Annual Symposium on Foundations of Computer Science, FOCS*, pages 436–453, 1994.

[Mil76] Gary L Miller. Riemann's hypothesis and tests for primality. *Journal of computer and system sciences*, 13(3):300–317, 1976.

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In *Advances in Cryptology - CRYPTO*, pages 620–649, 2019.

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In *10th Innovations in Theoretical Computer Science Conference, ITCS*, pages 60:1–60:15, 2019.

[Rab80] Michael O Rabin. Probabilistic algorithm for testing primality. *Journal of number theory*, 12(1):128–138, 1980.

[Rot21] Lior Rotem. Simple and efficient batch verification techniques for verifiable delay functions. In *TCC (3)*, volume 13044 of *Lecture Notes in Computer Science*, pages 382–414. Springer, 2021.

[RRR21] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. *SIAM J. Comput.*, 50(3), 2021.

[RS20] Lior Rotem and Gil Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In *Advances in Cryptology - CRYPTO*, pages 481–509, 2020.

[RSS20] Lior Rotem, Gil Segev, and Ido Shahaf. Generic-group delay functions require hidden-order groups. In *Advances in Cryptology - EUROCRYPT*, pages 155–180, 2020.

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. 1996.

[Sha90] Adi Shamir. Ip=pspace. In *31st Annual Symposium on Foundations of Computer Science, FOCS*, pages 11–15, 1990.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

[Sho06] Victor Shoup. *A computational introduction to number theory and algebra.* Cambridge University Press, 2006.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC*, pages 1–18, 2008.

[Wes20] Benjamin Wesolowski. Efficient verifiable delay functions. *J. Cryptol.*, 33(4):2113–2147, 2020.

[Zha22] Mark Zhandry. To label, or not to label (in generic groups). *IACR Cryptol. ePrint Arch.*, page 226, 2022.