RAD HAZU. MATEMATIČKE ZNANOSTI
Vol. 25 = 546 (2021): 15-31
DOI: `https://doi.org/10.21857/y54jofkjqm`

# A NOTE ON LOW ORDER ASSUMPTIONS IN RSA GROUPS

István András Seres and Péter Burcsi

Abstract. In this short note, we show that substantially weaker Low Order assumptions are sufficient to prove the soundness of Pietrzak's protocol for proof of exponentiation in groups of unknown order. This constitutes the first step to a better understanding of the asymptotic computational complexity of breaking the soundness of the protocol. Furthermore, we prove the equivalence of the (weaker) Low Order assumption(s) and the Factoring assumption in RSA groups for a non-negligible portion of moduli. We argue that in practice our reduction applies for a considerable amount of deployed moduli. Our results have cryptographic applications, most importantly in the theory of recently proposed verifiable delay function constructions. Finally, we describe how to certify RSA moduli free of low order elements.

## 1. Introduction

Verifiable delay functions (VDF) are powerful cryptographic tools [BBBF18] that opened up a plethora of applications, such as non-interactive timestamping [LSS19], proof of replication [FBGB19] or randomness beacons [BGB17]. A VDF is a function whose evaluation takes $\Omega(T)$ sequential steps and cannot be sped up by parallelism. Additionally, a prover, or evaluator, can produce publicly verifiable and succinct proofs that the function evaluation was correct. A crucial requirement for a VDF that there needs to be an exponential gap between function evaluation and proof verification time, more precisely verification time should be in $\mathcal{O}(\log T)$. Naturally, we require correctness and soundness from the applied proof systems. Specifically, an honest prover should always be able to convince the verifier, while a malicious prover should only be able to produce correct proofs with negligible probability.

Recent VDF constructions [Pie18, Wes19] proposed by Pietrzak and Wesolowski instantiate VDFs in groups of unknown order, i.e. groups for which the order cannot be computed efficiently [RSA78]. The existence of verifiable delay functions in the random oracle model is ruled out [MSW19],

2020 *Mathematics Subject Classification.* 94A60.

*Key words and phrases.* Verifiable delay functions, Low Order assumption, RSA.

moreover, groups of unknown order are shown to be mandatory for generic group delay functions [RSS20]. Both constructions [Pie18, Wes19] rely on novel, non-standard cryptographic assumptions. The soundness of these constructions can be proved by assuming the Low Order (LO) or Adaptive Root (AR) assumptions in groups of unknown order. Therefore there is an emerging need to understand better these new, non-standard cryptographic assumptions. In this note, we turn our attention to the LO assumption as it is a potentially weaker assumption than the AR assumption [BBF18].

**Our contributions.** In this note, we provide the following contributions.

- We observe that for the soundness of Pietrzak's proof of exponentiation succinct argument, one can assume substantially weaker LO assumptions than as previously defined in [BBF18]. In other words, we show that potentially it is harder to break soundness of Pietrzak's argument than as it was argued in [BBF18].
- We prove the equivalence of the LO and Factoring assumptions in RSA groups for a non-negligible portion of moduli. We argue that this result has practical consequences and that in practice one can deem the LO assumption to be equivalent to Factoring for the majority of used RSA moduli.
- We show how one could certify RSA moduli being free of low order elements using a non-interactive honest-verifier zero-knowledge proof system by Goldberg et al. [GRSB19].

The rest of this note is organized as follows. In Section 2 we provide background on the recently introduced LO and AR assumptions. We show the sufficiency of weaker LO assumptions in Section 3. In Section 4 we provide our reduction from Factoring to LO assumption for non-negligible RSA moduli. We describe a method to certify RSA moduli free of low order elements in Section 5. Finally, we point out open problems in Section 6.

## 2. Preliminaries

2.1. *Notations.* Let $\mathbb{G}$ be a group of unknown order. As usual, $\lambda$ denotes the security parameter. Integers denoted as $p, q$ are always primes, while $N$ is a semiprime, i.e. $N = pq$, sometimes referred to as RSA-modulus. All logarithms have two as their base, unless stated otherwise. The $\phi(\cdot)$ denotes Euler's totient function. Let $ord_m(a)$ denote the order of element $a$ in $\mathbb{Z}_m^{\times}$. In the following we assume that the size of the moduli is bounded by polynomial of the security parameter $s(\lambda)$, such that $p, q, \phi(N), N \approx \mathcal{O}(2^{s(\lambda)})$. We mean by $x \xleftarrow{\$} S$, that $x$ is uniformly at random sampled from set $S$.

2.2. *Proof of exponentiation and Pietrzak's succinct argument.* In a proof of exponentiation protocol in $\mathbb{G}$ the prover wants to convince the verifier that $h = g^{(2^T)}$ holds in $\mathbb{G}$. That is, the protocol is an argument system for the relation

$$\mathcal{L}_{\mathsf{EXP}} = \left\{ \left( \mathbb{G}, g, h, T \right) : h = g^{(2^T)} \in \mathbb{G} \right\}. \tag{2.1}$$

Pietrzak's proof system works as follows [Pie18].

1. The verifier checks that $g, h \in \mathbb{G}$ and outputs *reject* if not.
2. If $T = 1$, the verifier checks that $h = g^2$ in $\mathbb{G}$, outputs *accept* or *reject*, and stops.
3. If $T > 1$, the prover and verifier do:
   (a) The prover computes $v \leftarrow g^{(2^{T/2})} \in \mathbb{G}$ and sends $v$ to the verifier. The verifier checks that $v \in \mathbb{G}$ and outputs *reject* and stops, if not.
   Next, the prover needs to convince the verifier that $h = v^{(2^{T/2})}$ and $v = g^{(2^{T/2})}$, which proves that $h = g^{(2^{T/2})}$. Since the same exponent is used in both equalities, they can be verified simultaneously by checking a random linear combination, namely that
   $$v^r h = (g^r v)^{(2^{T/2})}, \quad \textit{where} \quad r \xleftarrow{\$} \{1, \ldots, 2^\lambda\}.$$
   The verifier and prover do so as follows.
   (b) The verifier sends the prover a random $r \xleftarrow{\$} \{1, \ldots, 2^\lambda\}$.
   (c) Both the prover and verifier compute $g_1 \leftarrow g^r v$ and $h_1 \leftarrow v^r h \in \mathbb{G}$.
   (d) The prover and verifier recursively engage in an interactive proof that $(\mathbb{G}, g_1, h_1, T/2) \in \mathcal{L}_{\mathsf{EXP}}$, namely that $h_1 = g_1^{(2^{T/2})} \in \mathbb{G}$.

FIGURE 1. Pietrzak's succinct argument for the proof of exponentiation language $\mathcal{L}_{\mathsf{EXP}}$, verbatim from [BBF18]

For simplicity, we assume that $T$ is a power of two in which case the protocol takes $\log T$ rounds. The protocol can be adjusted to handle arbitrary $T$, including a $T$ that is not a power of two [Pie18].

Naturally, the protocol can be made non-interactive by using the Fiat-Shamir heuristic. The prover generates the challenge $r$ at every level of the recursion by hashing the quantities $(\mathbb{G}, g, h, T, v)$ at that level, and appends $v$ to the overall proof $\pi$. Hence, the overall proof $\pi$ contains $\log T$ elements in $\mathbb{G}$. Proof generation has a complexity of $\frac{2T}{s\sqrt{T}}$ with $s$ being the amount of processors. At every level of the recursion, the verifier does two small exponentiations in $\mathbb{G}$ to compute $g_i$ and $h_i$ for the $i$th level of the recursion. Therefore, verifying the proof takes $\mathcal{O}(\log T)$ small exponentiations in $\mathbb{G}$.

An implementation study of Pietrzak's and Wesolowski's VDF construction showed that it is faster to verify Pietrzak's VDF than that of Wesolowski. However, Pietrzak's VDF comes with larger proofs, hence demanding slightly larger bandwidth [AVD20].

2.3. *RSA assumptions.* Informally, the LO assumption states that it is computationally infeasible to find a low order element in a random RSA group. Let us recall the formal definition of the LO assumption [BBF18].

DEFINITION 2.1. The Low Order assumption *holds for GGen if there is no efficient adversary $\mathcal{A}$ finding any element of low order:*

$$\Pr\left[u^l = 1,\ u \notin \{1,-1\} : \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (u,l) \leftarrow \mathcal{A}(\mathbb{G}) \\ \text{and } l < 2^{poly(\lambda)} \end{array}\right] \leq \text{negl}(\lambda). \tag{2.2}$$

We remark that the LO assumption unconditionally holds in $QR_N$, the group of quadratic residues $\bmod N$, since it contains no elements of low order. In case of RSA groups, we model $GGen(\lambda)$ as uniformly randomly sampling primes from an appropriate domain with respect to $\lambda$.

DEFINITION 2.2. The Factoring assumption *states that for random primes $p, q$ it is difficult to factor $N = pq$.*

It is trivial to see, that if there was an adversary $\mathcal{A}$ breaking the *Factoring* assumption, then one could easily calculate any roots $\bmod N$ by applying the Chinese Remainder Theorem. Computing arbitrary roots enables an adversary finding low order elements. Note that it is also true that technically, having a factoring algorithm that works for a non-negligible portion of RSA moduli, does not imply that the LO assumption is broken. It could be the case that factoring works if and only if the modulus is the product of two safe primes. For these moduli, we do not have low order elements at all. Certainly, what one implicitly understands by saying that the LO assumption is stronger than the factoring assumption is that a reduction exists for *GGen* that only outputs moduli with low order elements.

Even though we did not introduce Wesolowski's succinct argument for $\mathcal{L}_{\mathsf{EXP}}$, for sake of self-containedness we introduce the assumption needed to prove its soundness.

DEFINITION 2.3. The Adaptive Root assumption *holds for GGen if there is no efficient adversary $(\mathcal{A}_0, \mathcal{A}_1)$ that succeeds in the following task. First, $\mathcal{A}_0$ outputs an element $w \in \mathbb{G}$ and some state $st$. Then, a random prime $l$ in* Primes$(\lambda)$ *is chosen and $\mathcal{A}_1(w, l, st)$ outputs $w^{1/l} \in \mathbb{G}$. For all efficient*

$(\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\left[u^l = w \neq 1 : \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (w, st) \leftarrow \mathcal{A}_0(\mathbb{G}) \\ l \xleftarrow{\$} \Pi_\lambda = \text{Primes}(\lambda) \\ u \leftarrow \mathcal{A}_1(w, l, st) \end{array}\right] \leq \text{negl}(\lambda). \tag{2.3}$$

We note that the number of primes in $\Pi_\lambda$ should be exponential in $\lambda$: it is possible to precompute $w$ using $2^{|\Pi_\lambda|}$ exponentiations. Then, an adversary with $2^M$ memory can store intermediate exponents and compute adaptive roots using $2^{|\Pi_\lambda|-M}$ exponentiations for each. Moreover, it was shown that the Adaptive Root assumption holds in the generic group model [BBF19].

2.4. *Number theoretic tools.* We recall the following lemma without proof.

Lemma 2.4. *The map $x \to x^e$ mod $N$ is a permutation of $Z_N^*$ if and only if* $\gcd(e, \phi(N)) = 1$.

Furthermore, let us define the language of RSA public keys $(N, e)$, such that the map $x \to x^e \bmod N$ is a permutation over $Z_N^*$:

$$\mathcal{L}_{\mathsf{perm}\mathbb{Z}_N^*} = \{(N, e) | N, e > 0 \wedge \gcd(e, \phi(N)) = 1\}. \tag{2.4}$$

For this particular language, Goldberg et al. devised a public-coin protocol [GRSB19] with perfect completeness, perfect honest-verifier zero-knowledge, and statistical soundness.

Later we will need to count the number of integers without factors from an interval. Therefore we introduce the following function and notation.

Definition 2.5. *Denote by $\Gamma(x, y, z)$ the number of all positive integers less than $x$ which are free of prime divisors from the interval $(z, y]$.*

Theorem 2.6 (Weingartner [Wei01]). *Let $u = \frac{\log x}{\log y}$ and $v = \frac{\log x}{\log z}$. Then uniformly for $\frac{3}{2} \leq z \leq y$, whenever $x \geq yz$, we have the following asymptotic relationship for $\Gamma(x, y, z)$:*

$$\Gamma(x, y, z) = x\eta(u, v)\Big(1 + \mathcal{O}\Big(\frac{1}{\log z}\Big)\Big), \tag{2.5}$$

*where $1 \leq u \leq v$ and $\eta(u, v) \geq \frac{u}{2v}$.*

*We remark that a similar result and asymptotic was obtained by Warlimont [War90], however, he solely proved his results for fixed $z$. We will crucially rely on the uniform convergence of the asymptotics in Equation (2.5).*

In the following we apply theorems about the cycle structure of repeated exponentiation in $\mathbb{Z}_p$.

Theorem 2.7 (Chou and Shparlinski [CS04]). *The cycle length of the map $u \mapsto u^e \bmod p$ for a purely periodic element $u$ is $ord_{ord_p u} e$.*

One can easily generalize the result in Theorem 2.7 for composite moduli using the Chinese Remainder Theorem. Namely, for composite moduli the cycle length of a purely periodic element $u$ under the map $u \mapsto u^e \bmod N = pq$ is $ord_{ord_N u} e$ [BG98].

## 3. Soundness of Pietrzak's argument and weaker LO assumptions

Boneh et al. introduced the low order assumption as a sufficient and necessary assumption to prove the soundness of Pietrzak's argument [BBF18]. In this section, we show that the original definition of the LO assumption, cf. Section 2.3, is not necessary for proving soundness. We will show that one needs to assume almost exponentially weaker assumptions as a necessary and sufficient assumption for soundness of Pietrzak's proof of exponentiation protocol.

### 3.1. *(Non)-necessity of the LO assumption and weaker LO assumptions.*

Let us assume that the LO assumption, cf. Definition 2.1, is broken. What is the probability that such a potent adversary could break the soundness of Pietrzak's argument system? As it turns out, it can still be negligible.

The main observation in the soundness analysis is that whenever a malicious prover finds $(u, l)$, a low order element $u$, with order $l < 2^{poly(\lambda)}$ can potentially break the soundness of the argument system. This is because if $(\mathbb{G}, g, h, T) \in \mathcal{L}_{\mathsf{EXP}}$, then $(\mathbb{G}, g, hu, T) \notin \mathcal{L}_{\mathsf{EXP}}$ and will be incorrectly accepted by the verifier with probability $1/l$. Towards breaking soundness malicious prover sends $v \leftarrow g^{(2^{T/2})}u \in \mathbb{G}$. Soundness of the argument system does not hold whenever $r + 1 \equiv 2^{T/2} \pmod{l}$, since $(\mathbb{G}, g^r v, v^r(hu), T/2) \in \mathcal{L}_{\mathsf{EXP}}$.

However, there might be an adversary $\mathcal{A}$ breaking the LO assumption with non-negligible probability, even though their success probability in breaking the soundness of the argument system is negligible. This can occur, if $\mathcal{A}$ is only able to find low order elements with their order in $2^{\Theta(poly(\lambda))}$. In this case the probability that $\mathcal{A}$ breaks soundness is at most $1/2^{\Theta(poly(\lambda))}$, hence negligible.

After this discussion, one can see that the $2^{poly(\lambda)}$ upper bound for the order of the low order element needs to be decreased to get the weakest necessary assumption for proving soundness of Pietrzak's argument. Therefore, in quest to define a sufficiently weak LO assumption with the lowest permissible bound on the order of the low order element, we introduce the following smallest subexponential LO assumption.

Definition 3.1. The Subexponential Low Order assumption. *For any probabilistic polynomial time adversary $\mathcal{A}$, and for any $0 < \epsilon$, finding any* *element of subexponentially low order is hard:*

$$\Pr\left[ u^l = 1,\ u \notin \{1,-1\} : \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (u,l) \leftarrow \mathcal{A}(\mathbb{G}) \\ \text{and } l < 2^{\log^{1+\epsilon}(\lambda)} \end{array} \right] \leq \text{negl}(\lambda). \tag{3.1}$$

We note, that the $2^{\log^{1+\epsilon}(\lambda)}$ upper bound cannot be lowered to a polynomial-bound as then the assumption would not be sufficient, see next subsection. In a nutshell, for sufficiency to hold one needs to assume a super-polynomial upper bound for the order of the low order element. Furthermore, we remark that even the weaker LO assumption introduced in Definition 3.1 is not necessary, because there are non-negligible RSA-moduli with $\phi(N)$ having divisors between any $poly(\lambda)$ and $2^{\log^{1+\epsilon}(\lambda)}$ [BS13]. Again, there might be adversaries who are only able to find low order elements with their order being between $poly(\lambda)$ and $2^{\log^{1+\epsilon}(\lambda)}$, therefore their success probability in breaking soundness of Pietrzak's argument would be negligible.

3.2. *Sufficiency of weaker LO assumptions for soundness of Pietrzak's argument.* Let $f(\lambda)$ denote the function limiting the maximal order of the low order element in an LO assumption. We see that in Definition 2.1, $f(\lambda) = 2^{poly(\lambda)}$, while in Definition 3.1, $f(\lambda) = 2^{\log^{1+\epsilon}(\lambda)}$. We are interested in finding the minimal $f(\lambda)$ such that the LO assumption with that $f(\lambda)$ is still sufficient for proving the soundness of Pietrzak's argument system for $\mathcal{L}_{\mathsf{EXP}}$.

THEOREM 3.2. *If the soundness of Pietrzak's succinct argument for proof of exponentiation is broken, then so is the subexponential LO assumption, cf. Definition 3.1.*

PROOF. Hereby we reuse the proof given by Boneh et al. [BBF18] with modifications to our specific setting. Hence, we recall their proof for the sufficiency of the LO assumption for breaking the soundness of Pietrzak's argument. Let $\mathcal{A}$ be an adversary who breaks the soundness of Pietrzak's argument with non-negligible probability $\epsilon$. We use a forking argument to construct an adversary $\mathcal{B}$ that breaks the low order assumption using $\mathcal{A}$.

Recall that $2^t$ is an upper bound on the value $T$ output by $\mathcal{A}$. Let $\mathcal{A}(\mathbb{G}, r_0, \ldots, r_{t-1}; R)$ denote an execution of $\mathcal{A}$ with random tape $R$, where $r_0, \ldots, r_{t-1}$ are the verifier's challenges at each level of the recursion. The adversary $\mathcal{A}$ outputs $(J, \sigma)$. The protocol transcript is denoted by $\sigma$ which is a sequence of $t+1$ tuples: $\sigma = (P_0, v_0), \ldots, (P_t, v_t)$, where $P_i = (\mathbb{G}, g_i, h_i, T/2^i)$ is the input to the recursion at level $i$, and $v_i$ is the prover's message at level $i$. Adversary $\mathcal{A}$ also outputs the smallest index $J$ for which $P_J \notin \mathcal{L}_{\mathsf{EXP}}$ but $P_{J+1} \in \mathcal{L}_{\mathsf{EXP}}$ whenever $P_0 \in \mathcal{L}_{\mathsf{EXP}}$ and $P_t \notin \mathcal{L}_{\mathsf{EXP}}$. Otherwise $J = -1$ if such an index does not exist[1]. Recall that $g_i \leftarrow g_{i-1}^{r_{i-1}} v_{i-1}$ and $h_i \leftarrow v_{i-1}^{r_{i-1}} h_{i-1}$ for $i = 1, \dots, t$. Here we assume $T = 2^t$, but if $T < 2^t$ then we replicate the last pair $(P_{\log T}, v_{\log T})$ to get a full transcript of $t+1$ tuples.

1. Let $K = [1, 2^{\log^{1+\epsilon}(\lambda)}]$.
2. Input generator $\mathsf{IG}$ samples $\mathsf{x} = (x_0, x_1, \dots, x_{t-1}) \overset{\$}{\leftarrow} [0, 2^{\lambda - 2^{\log^{1+\epsilon}(\lambda)}}]$.
3. We define algorithm $\mathcal{A}'$ on random tape $R$ and $k_i \in K$ for all $i$, as follows: $\mathcal{A}'(\mathsf{x}, k_0, k_1, \dots, k_{t-1}; R)$ invokes $\mathcal{A}(\mathbb{G}, r_0, \dots, r_{t-1}; R)$ with $r_i = x_i \cdot 2^{\log^{1+\epsilon}(\lambda)} + k_i$. Algorithm $\mathcal{A}'$ returns the same output as $\mathcal{A}$. Note that whenever $k_i \overset{\$}{\leftarrow} K$ and $\mathsf{x} \overset{\$}{\leftarrow} \mathsf{IG}$, then $r_i$ is also sampled uniformly at random from $[1, 2^\lambda]$. Hence, the success probability of $\mathcal{A}'$ equals that of $\mathcal{A}$.
4. Next, define the following probabilistic experiment $\mathsf{F}_{\mathcal{A}'}(\mathsf{x})$: let $P_0 \notin \mathcal{L}_{\mathsf{EXP}}$ but $P_t \in \mathcal{L}_{\mathsf{EXP}}$ (i.e., the verifier incorrectly accepts $P_0$) then:
    - choose a random tape $R$ for $\mathcal{A}'$.
    - Sample $k_i \overset{\$}{\leftarrow} K$ for $i \in [0, t-1]$.
    - We obtain $(I, \sigma) \leftarrow \mathcal{A}'(\mathsf{x}, k_0, k_1, \dots, k_{t-1}; R)$.
    - If $I = -1$, output *fail*.
    - If $I \geq 0$, then sample new $k_i' \overset{\$}{\leftarrow} K$ for $i = (I+1, \dots, t-1)$.
    - We obtain $(I', \sigma') \leftarrow \mathcal{A}'(\mathsf{x}, k_0, \dots, k_I, k_{I+1}', \dots, k_{t-1}'; R)$
    - If $I = I' \wedge k_{I+1} \neq k_{I+1}'$, then return $(I, \sigma, \sigma')$ and *success*.
    - Else return *fail*.

Let $\mathcal{E}$ be the event that $\mathsf{F}_{\mathcal{A}'}(\mathsf{x})$ outputs *success*. By applying the general forking lemma by Bellare and Neven [BN06] we have that $\mathcal{E}$ happens with probability $(\epsilon^2/t) - (\epsilon / 2^{\log^{1+\epsilon}(\lambda)})$. This probability is non-negligible, whenever $\epsilon$ is non-negligible.

Now we establish why event $\mathcal{E}$ produces low order element for adversary $\mathcal{B}$. When $\mathcal{E}$ happens, we have $P_I \notin \mathcal{L}_{\mathsf{EXP}}$ and $P_{I+1}, P_{I+1}' \in \mathcal{L}_{\mathsf{EXP}}$. Therefore, if $\mathsf{F}_{\mathcal{A}'}(\mathsf{x})$ outputs $(I, \sigma, \sigma')$, adversary $\mathcal{B}$ obtains the sextuple $(g, h, \hat{T}, v, r, r')$ with the following properties.

$$h \neq h^{(2^{2T})} \quad \text{and} \quad (g^r v)^{(2^{\hat{T}})} = v^r h \quad \text{and} \quad (g^{r'} v)^{(2^{\hat{T}})} = v^{r'} h. \tag{3.2}$$

Re-arranging terms of the two equalities on the right we get

$$(g^{(2^{\hat{T}})}/v)^r = h/v^{(2^{\hat{T}})} \quad \textit{and} \quad (g^{(2^{\hat{T}})}/v)^{r'} = h/v^{(2^{\hat{T}})}. \tag{3.3}$$

[1]Outputting $J \overset{\$}{\leftarrow} [-1, t-1]$ only reduces adversary's success probability with a factor of $t+1$.

Dividing the left equality by the right we obtain

$$(g^{(2^{\hat{T}})}/v)^{r-r'} = 1. \tag{3.4}$$

Let $u := g^{(2^{\hat{T}})}/v$. Next we establish an upper bound on the order of $u \in \mathbb{G}$. It is guaranteed that $r \neq r'$, since $k_{I+1} \neq k'_{I+1}$. We observe that the order of $u$ can be bounded by:

$$\begin{aligned}|r - r'| &= |(x_{I+1} \cdot 2^{\log^{1+\epsilon}(\lambda)} + k_{I+1}) - (x_{I+1} \cdot 2^{\log^{1+\epsilon}(\lambda)} + k'_{I+1})| \\ &= |k_{I+1} - k'_{I+1}| \leq 2^{\log^{1+\epsilon}(\lambda)}\end{aligned}$$

Hence, we conclude that $(u, r - r')$ is a pair which breaks the subexponential low order assumption in $\mathbb{G}$. □

We remark that one could replace $f(\lambda) = 2^{\log^{1+\epsilon}(\lambda)}$ with any superpolynomial function of $\lambda$ in Theorem 3.2.

## 4. Partial reductions of Factoring to the LO assumption

In this section we provide two partial reductions, cf. Section 4.1 and 4.2, of the Factoring assumption to the Low Order assumption. In Section 4.3 we show that for the vast majority of deployed moduli finding a low order element is no easier than factoring.

### 4.1. *Partial reduction for low order smooth integers.*

In this section we prove the equivalence of the LO and the Factoring assumption for a noticeable fraction of the moduli. We use the LO assumption introduced by Boneh et al. [BBF18], cf. Section 2.3. However, the proof enclosed hereby would equally work well for the weaker, subexponential variants of the LO assumption. Specifically, in the following theorem we assume that the RSA-modulus $N$ is generated in a way such that $\phi(N)$ has no prime factor in $(\mathfrak{B}, 2^{poly(\lambda)}]$ for a constant $\mathfrak{B}$. We call these moduli as low order smooth integers. We note that the factor base $\mathfrak{B}$ could be bounded by a polynomial of $\lambda$, instead of a constant $\mathfrak{B}$. However, for ease of exposition we claim reduction only with a constant $\mathfrak{B}$. Additionally, we assume that $\gcd(p-1, q-1) = 2$. In Section 4.3 we argue that in practice the majority of RSA moduli satisfy these requirements in a typical parameter setting.

Theorem 4.1. *Let $\mathcal{B}$ be a fixed integer. The Factoring assumption is reducible in polynomial time to the Low Order assumption for RSA-moduli when $\phi(N)$ has no prime factor between $\mathfrak{B}$ and $2^{poly(\lambda)}$ and* $\gcd(p-1, q-1) = 2$.

Proof. Let us assume there exists an efficient adversary $\mathcal{A}$, who can break the LO assumption with non-negligible probability. Express differently, there exists a polynomial $q(\lambda)$, such that

$$\Pr[\mathcal{A} \quad \text{breaks} \quad \text{LO}] \geq \frac{1}{q(\lambda)}. \tag{4.1}$$

We devise an efficient adversary $\mathcal{B}$ who can factor non-negligible fraction of random RSA moduli by using $\mathcal{A}$ as a subroutine. Adversary $\mathcal{B}$ operates as follows. Upon receiving a random semiprime $N$ it invokes $\mathcal{A}$ on $\mathbb{Z}_N^*$. By our assumption, adversary $\mathcal{A}$ with non-negligible probability outputs a pair $(u, l)$ such that $u^l \equiv 1 \pmod N$ and $2 \leq l \leq 2^{poly(\lambda)} \wedge u \neq -1$. Note that, the order $l$ of $u \in (\mathbb{Z}/pq\mathbb{Z})^\times \cong (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times$ is the least common multiple of its (multiplicative) orders modulo $p$ and modulo $q$, i.e. $l = \operatorname{lcm}(ord_p(u), ord_q(u))$.

Note that, whenever $ord_p(u) \neq ord_q(u)$, adversary $\mathcal{B}$ could factor $N = pq$ if $l$ was smooth enough. The reason being that, adversary $\mathcal{B}$ raises $u$ to the power of $\frac{l}{r}$ for all prime factors $r$ of $l$, until modulo one prime factor of $N$, but not the other the order of $u$ divides $\frac{l}{r}$. This can be detected by $0 < \gcd(u^{\frac{l}{r}} - 1 \bmod N, N) < N$, hence factoring the modulus $N$. In our reduction adversary $\mathcal{B}$ tries to find all prime factors of $l$ in order to find a non-trivial factor of $N$ as described above. This will be the technique employed by adversary $\mathcal{B}$. Hence, towards our goal one thing that we need to show is that $ord_p(u) \neq ord_q(u)$ with non-negligible probability.

First, let us assess the probability when $ord_p(u) = ord_q(u)$ for randomly chosen primes $p, q$. This probability needs to be established as in this case one cannot factor $N$ with the aforementioned technique. We show that $\gcd(\frac{p-1}{2}, \frac{q-1}{2}) = 1$ with constant probability. Since $p$ and $q$ were chosen uniformly random, also $(p-1)/2$ and $(q-1)/2$ behave almost like random integers if we consider their divisibility by other primes (there is one excluded residue class for each prime). The probability of coprimality of random integers is $\frac{1}{\zeta(2)} = \frac{6}{\pi^2} \approx 0.61$. In our case, we have to adjust this value because of the excluded classes, but we still have that $\gcd(p-1, q-1) = 2$ with constant positive probability, therefore whenever $ord_p(u) = ord_q(u)$, then this quantity is either 1 or 2. We examine these two cases in more detail.

- $ord_p(u) = ord_q(u) = 1$. This is only possible if $u = 1$, which cannot be the case by the definition of the LO assumption, see Definition 2.1.
- $ord_p(u) = ord_q(u) = 2$. In this case $ord_N(u) = \operatorname{lcm}(ord_p(u), ord_q(u)) = 2$. Since $u \notin \{1, -1\}$, therefore $u$ is another non-trivial square root of 1. This also implies that one can factor $N$ as $pq = N | u^2 - 1 = (u-1)(u+1) \wedge u \notin \{1, -1\}$. Therefore $\gcd(N, u-1) = p \vee \gcd(N, u+1) = p$.

Hence, we can conclude that for randomly chosen primes $p, q$ with constant probability $ord_p(u) \neq ord_q(u)$.

Second, the goal of adversary $\mathcal{B}$ is to obtain all the prime factors of $l$. For that end, adversary chooses a constant bound $\mathfrak{B}$, say $2^{10}$. Hence, the adversary would like $l$ to be a $\mathfrak{B}$-smooth integer in order to be able to factor it efficiently. Whenever adversary receives order $l$ $(1 \leq l \leq 2^{poly(\lambda)})$ of $u$, then adversary would like to find all of its prime factors in a brute force-manner, but still in polynomial-time in $\lambda$. Namely, adversary $\mathcal{B}$ wants to find $l$'s smallest prime factor $l_1$, which is smaller than $\mathfrak{B}$. Suppose $a_1$ is the largest integer such that $l_1^{a_1} | l$. Then, recursively we would like to find the smallest prime factor of $\frac{l}{l_1^{a_1}}$, denoted $l_2$ which is smaller than $\mathfrak{B}$ and so on.

This approach succeeds whenever $\mathcal{B}$ receives a pair $(u, l)$ from $\mathcal{A}$, where $l$ is $\mathfrak{B}$-smooth. This can be guaranteed if $\phi(N)$ has no prime factors between $\mathfrak{B}$ and $2^{poly(\lambda)}$. Therefore, we compute now the fraction of those primes up to $N$, that do not have prime factors between $\mathfrak{B}$ and $2^{poly(\lambda)}$. We need to establish the fraction of primes up to $N = \mathcal{O}(2^{s(\lambda)})$, that do not have prime factors in $(\mathfrak{B}, 2^{poly(\lambda)}]$. Let us call these integers as low order smooth integers, additionally let $\mathcal{P}_{los}$ be the probability that a randomly chosen integer is low order smooth. We obtain the following asymptotic by applying Equation (2.5):

$$\begin{aligned} \mathcal{P}_{los}(\lambda) &= \frac{\Gamma(2^{s(\lambda)}, 2^{poly(\lambda)}, \mathfrak{B})}{2^{s(\lambda)}} \approx \frac{2^{s(\lambda)} \eta(s(\lambda)/poly(\lambda), s(\lambda)/\mathfrak{B})}{2^{s(\lambda)}} \\ &\geq \frac{s(\lambda)/poly(\lambda)}{2s(\lambda)/\mathfrak{B}} = \frac{\mathfrak{B}}{2poly(\lambda)}. \end{aligned} \tag{4.2}$$

Hence, we established that $\mathcal{P}_{los}$ is non-negligible in $\lambda$. It follows that the probability that the order of a random RSA-modulus does not have a prime factor in $(\mathfrak{B}, 2^{poly(\lambda)}]$ is $\mathcal{P}_{los}^2$, i.e. non-negligible. We can establish now the success probability of adversary $\mathcal{B}$ breaking the Factoring assumption:

$$\Pr[\mathcal{B} \text{ breaks Factoring}] \geq \frac{6}{\pi^2} q(\lambda) \mathcal{P}_{los}^2(\lambda). \tag{4.3}$$

Therefore, we conclude our proof that the success probability of the efficient adversary $\mathcal{B}$ is non-negligible. ∎

We remark that in the reduction we could have allowed $\phi(N)$ to have a single prime factor in $(\mathfrak{B}, 2^{poly(\lambda)}]$. Once $\mathcal{B}$ factors out all the prime factors smaller than $\mathfrak{B}$ from the low order $l$, adversary $\mathcal{B}$ can establish in probabilistic polynomial time whether the resulting integer is a prime power. If yes, then also in those cases $\mathcal{B}$ can factor efficiently the low order $l$.

4.2. *Another partial reduction using the generalized cycling attack.* In this section, we give a different reduction for Theorem 4.1 (Note that Theorem 4.2 is an equivalent restatement of Theorem 4.1). The high-level idea is that an efficient low order adversary would be able to efficiently launch the generalized cycling attack [SN77, Ber82] to factor RSA moduli such that $\phi(N)$ has no prime factor between a constant $\mathfrak{B}$ (or equivalently a polynomial of $\lambda$) and $2^{\log^{1+\epsilon}(\lambda)}$, for $\epsilon > 0$ and $\gcd(p-1, q-1) = 2$.

This result might suggest that these two aforementioned conditions about the modulus (low order smoothness and coprimality of $\frac{p-1}{2}$ and $\frac{q-1}{2}$) are not arbitrary or incidental. It might imply that here, there is a barrier in reducing the Factoring assumption to the Low Order assumption. Therefore, in practice it might be useful or even necessary to prove that a modulus is safe against low order adversaries. For that end, in Section 5, we describe an application of a zero-knowledge proof system to certify RSA moduli being free of low order elements.

Theorem 4.2. *Let $t(\cdot)$ be a polynomial. The Factoring assumption is reducible in polynomial time to the Low Order assumption for RSA-moduli when $\phi(N)$ has no prime factor in $(t(\lambda), 2^{\log^{1+\epsilon}(\lambda)}]$, for $\epsilon > 0$ and $\gcd(p-1, q-1) = 2$.*

Proof. Suppose, there exists an efficient low order adversary $\mathcal{A}$ that outputs a pair $(u, l)$ with non-negligible probability such that $u^l \equiv 1 \pmod N$ and $2 \leq l \leq 2^{\log^{1+\epsilon}(\lambda)}$. Express differently, there exists a polynomial $q(\lambda)$, such that

$$\Pr[\mathcal{A} \quad \text{breaks} \quad \text{LO}] \geq \frac{1}{q(\lambda)}. \tag{4.4}$$

We devise an efficient adversary $\mathcal{B}$ who can factor non-negligible fraction of random RSA moduli by using $\mathcal{A}$ as a subroutine. Adversary $\mathcal{B}$ operates as follows. Upon receiving a random semiprime $N$ it invokes $\mathcal{A}$ on $\mathbb{Z}_N^*$. By our assumption, adversary $\mathcal{A}$ with non-negligible probability outputs a pair $(u, l)$, i.e. a low order element $u$ with order $l$. In the following, we will take advantage of the generalized cycling attack on RSA groups [Ber82].

In the generalized cycling attack an adversary can factor the modulus by generating a sequence of "reencryptions" of the element $u$ with public exponent $e$. Namely adversary $\mathcal{B}$ generates the sequence $X = (u^e \bmod N, u^{e^2} \bmod N, \ldots, u^{e^k} \equiv u \bmod N)$ and hopes to find a factor of $N$ by computing $1 < \gcd(u^{e^i} - u, N) < N$ for all $1 \leq i \leq k$. The generalized cycling attack could succeed at latest when $\mathcal{B}$ finds a cycle, i.e. finds a $k$ such that $u^{e^k} \equiv u \pmod N$. The low order adversary $\mathcal{B}$ applies $u$ in the generalized cycling attack, since Theorem 2.7 by Chou and Shparlinski ensures that $u$ will have a small cycle length if its order is low.

Now, let us analyze when the generalized cycling attack will not yield a factorization. The sequence $X$ is eventually periodic, but might contain in the beginning an aperiodic part. The element $u$ is said to be purely periodic if $X$ does not have tail elements. Let $\pi_{\{X\},p}$ denote the period of the sequence $X \bmod p$. Furthermore let $e = p_1^{n_1} \cdot \dots \cdot p_s^{n_s}$ and $p - 1 = p_1^{r_1} \cdot \dots \cdot p_s^{r_s} \rho$, where $p_1, \ldots p_s$ are distinct primes and $\gcd(p_1, \ldots, p_s, \rho) = 1$. The generalized cycling attack does not yield a factorization, whenever the periods of the sequence $X \bmod p$ and $q$ are equal, i.e. $\pi_{\{X\},p} = \pi_{\{X\},q}$ [GS99]. To analyze this situation we make the following case distinction.

- $u$ is purely periodic: Let us assume that every element of $X$ is in the purely periodic part of the cycle. Let $d = ord_p(u)$ and $d' = ord_q(u)$. It is known that $\pi_{\{X\},p} = ord_d(e)$ [CS04]. Hence, not having equal cycle lengths is equivalent to $ord_d(e) \neq ord_{d'}(e)$. A necessary requirement of this is that $d = ord_p(u) \neq ord_q(u) = d'$.
- $u$ is a tail element: If $u$ is not in the purely periodic part of the cycle, then $d = ord_p(u)$ does not divide $\rho$ [CS04]. The tail lengths of $X$ modulo $p$ is a function of $ord_p(u)$ [VS04]. Therefore, whenever $ord_p(u) = ord_q(u)$, the tail lengths of $X$ modulo $p$ and $q$ are equal. But, this also signifies that the periodic parts will have the same length size.

In both cases we saw, that the generalized cycling attack does not produce a factorization, if $ord_p(u) = ord_q(u)$. This implies that the reduction is successful for the same type of moduli as stated in Theorem 4.1 and 4.2. We already showed that these moduli amount to a non-negligible portion of all random moduli in the proof of Theorem 4.1.

Lastly, we discuss the running time of the reduction and the success probability of the low order adversary $\mathcal{B}$.

The generalized cycling attack can be launched efficiently, if adversary $\mathcal{B}$ can find a small cycle length. To that end, $\mathcal{B}$ will use the pair $(u, l)$ received from the low order adversary $\mathcal{A}$. The cycle length $k$ of $u$ under the map $u \mapsto u^e \bmod N$ is $k = ord_l(e)$ [BG98]. However, $k$ can still be large, i.e. $k \in \Theta(l) \approx 2^{\log^{1+\epsilon}(\lambda)}$. Since $N$ does not have factors in $(t(\lambda), 2^{\log^{1+\epsilon}(\lambda)}]$, therefore $\mathcal{B}$ can factor $l$ by sieving up to $t(\lambda)$. Let $m$ be one of the factors of $l$ smaller than $t(\lambda)$. Then by setting $u := u^{l/m}$ we will have that $k = ord_m(e)$, i.e. in the generalized cycling attack every choice of exponent $e$ will produce a cycle length $k \leq m$, which is polynomial in $\lambda$. This implies that the generalized cycling attack, hence the reduction runs in polynomial time in $\lambda$.

Finally, we note, that the success probability of adversary $\mathcal{B}$ is at least that of the low order adversary $\mathcal{A}$. Namely, whenever a low order element is found, $\mathcal{B}$ can launch a generalized cycling attack with a polynomial cycle length $k$. $\square$

4.3. *Practical consequences of the reductions.* Hereby, we give an estimate on the portion of RSA moduli used in practice, for which the reductions presented in Section 4.1 and 4.2 guarantee an equal level of security for the LO problem, as for the classical Factoring problem. Let $\epsilon = 0.6$ and $\lambda = 80$, consequently, the size of the moduli should be 1024 bits to provide $\lambda$-bit security.

OpenSSL is an open-source cryptographic library, which is widely used and the most popular on the internet [ŠNS+16]. OpenSSL generates primes in a way that it ensures that no prime from 3 to 17863 divides $p-1$. Let $S$ denote the following set, $S = \{(p-1)/2 \,|\, p_i \nmid p-1, 2 \leq i \leq 2048\}$, where $p_i$ is the $i$th prime. Hence, the probability of coprimality for two randomly sampled integers from $S$ is $\lim_{n\to\infty} \prod_{i\geq 2049}^{n}(1-1/p_i^2) \approx 0.7499$. Hence we let $\mathfrak{B} = 17863$. The portion of integers having no prime factor between $\mathfrak{B}$ and $2^{\log^{1.6}(80)}$ amounts to $\frac{\log(\mathfrak{B})}{\log^{1.6}(80)} = 0.7389$ that can be obtained by using Weingartner's theorem, cf. Equation (2.5).

Let us consider the reductions presented in Section 4.1 and 4.2. Theorem 4.1 and 4.2 require that $\phi(N)$ does not have factors in $(\mathfrak{B}, 2^{\log^{1+\epsilon}(\lambda)}]$ and additionally, that $\gcd(p-1, q-1) = 2$. Hence, we have that the probability that an RSA modulus randomly generated by OpenSSL provides the same security guarantees for the LO assumption as for the Factoring is approximately $0.7389 \cdot 0.7499 = 0.5541$.

In summary, for the majority of RSA moduli used in practice, if one could find a low order element, then they would be also able to factor those moduli using the reductions introduced in Section 4.1 and 4.2.

## 5. Certifying RSA moduli free of low order elements

In certain use cases, e.g. in a public key infrastructure setting or for a VDF, it might be useful if users could prove that their RSA-modulus is free of low order elements. To that end, one could certify RSA moduli applying techniques developed by Goldberg et al. [GRSB19].

Specifically, we assume a user wants to prove in zero-knowledge that their RSA modulus $N$ is free of low order elements. Express differently, a user wants to show that $\phi(N)$ has no divisors smaller than a certain bound $\mathfrak{B}$. Let $p_n$ denote the largest prime smaller than $\mathfrak{B}$. Then let $e = \prod_{i=1}^{n} p_i$, where $p_i$ is odd prime, i.e. $e$ is the $n$th primorial divided by two. Consequently, there cannot be elements $\bmod N$ with order smaller than $\mathfrak{B}$ if and only if $\gcd(e, N) = 1$. Put differently, $N$ has no low order elements if and only if $x \to x^e \bmod N$ is a permutation. Applying the zero-knowledge proof system for the language $\mathcal{L}_{\mathsf{perm}\mathbb{Z}_N^*}$ this can be proved efficiently [GRSB19].

In a typical parameter setting (1024-bit RSA moduli, $2^{-80}$ soundness error for the certification, $\mathfrak{B} = 2^{10}$) the low order RSA public key certification would consist of 50 elements of $\mathbb{Z}_N$. Hence the size of the proof amounts to 6.4KB. Generating the certification costs roughly 50 full-length RSA exponentiations modulo $N$, hence it is also feasible to calculate the proof in a distributed RSA key generation scenario. Meanwhile, each verifier pays the one-time cost of verifying the certification, which is also roughly equal to 50 full-length exponentiations. We note that the number of elements consisting of the proof and number of exponentiations the verifier needs to compute depends only on the admitted soundness error of the proof system and not the size of the moduli. More precisely, both quantities are roughly $\approx \lambda/\log 3$.

## 6. Open Problems

It is a fascinating open problem to explore more connections between novel and standard RSA assumptions[2]. For instance, it would be fruitful to establish the relation of the Adaptive Root assumption [BBF18] and the (Strong) RSA assumption. The Adaptive Root assumption underpins the security of Wesolowski's VDF construction [Wes19] and several batching techniques proposed for RSA accumulators [BBF19].

Acknowledgements.

We thank Benjamin Wesolowski, Dankrad Feist, Dmitry Khovratovich and Justin Drake for insightful discussions. The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications).

## References

[AVD20] V. Attias, L. Vigneri and V. Dimitrov, *Implementation study of two verifiable delay functions*, in: 2nd International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2020), Open Access Series in Informatics (OASIcs) **80**, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021, pp. 9:1–9:14.

[BS13] E. Bach and J. P. Sorenson, *Approximately counting semismooth integers*, in: Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation, ACM, New York, 2013, pp. 23–30.

[BN06] M. Bellare and G. Neven, *Multi-signatures in the plain public-key model and a general forking lemma*, in: Proceedings of the 13th ACM conference on Computer and communications security, 2006, pp. 390–399.

[Ber82] S. Berkovits, *Factoring via superencyrption*, Cryptologia **6** (1982), 229–237.

[BBBF18] D. Boneh, J. Bonneau, B. Bünz and B. Fisch, *Verifiable delay functions*, in: Advances in Cryptology – CRYPTO 2018. Part I, Lecture Notes in Comput. Sci. **10991**, Springer, Cham, 2018, pp. 757–788.

[BBF18] D. Boneh, B. Bünz and B. Fisch, *A survey of two verifiable delay functions*, IACR Cryptology ePrint Archive 2018:712, 2018.

[BBF19] D. Boneh, B. Bünz and B. Fisch, *Batching techniques for accumulators with applications to IOPs and stateless blockchains*, in: Advances in Cryptology – CRYPTO 2019, Lecture Notes in Comput. Sci. **11692**, Springer, Cham, 2019, pp. 561–586.

[BG98] J. J. Brennan and B. Geist, *Analysis of iterated modular exponentiation: the orbits of $x^\alpha$ mod $n$*, Des. Codes Cryptogr. **13** (1998), 229–245.

[BGB17] B. Bünz, S. Goldfeder and J. Bonneau, *Proofs-of-delay and randomness beacons in ethereum*, IEEE Security and Privacy on the blockchain (IEEE S & B), 2017.

[CS04] W.-S. Chou and I. E. Shparlinski, *On the cycle structure of repeated exponentiation modulo a prime*, J. Number Theory **107** (2004), 345–356.

[FBGB19] B. Fisch, J. Bonneau, N. Greco and J. Benet, *Scaling proof-of-replication for filecoin mining*, Technical Report, Stanford University, May 2019.

[2]See: `https://rsa.cash/rsa-assumptions/`

[GRSB19] S. Goldberg, L. Reyzin, O. Sagga and F. Baldimtsi, *Efficient noninteractive certification of RSA moduli and beyond*, in: Advances in Cryptology – ASIACRYPT 2019, Lecture Notes in Comput. Sci. **11923**, Springer, Cham, 2019, pp. 700–727.

[GS99] M. Gysin and J. Seberry, *Generalised cycling attacks on RSA and strong RSA primes*, in: Australasian Conference on Information Security and Privacy, Lecture Notes in Comput. Sci. **1587**, Springer, Berlin, 1999, pp. 149–163.

[LSS19] E. Landerreche, M. Stevens and C. Schaffner, *Non-interactive cryptographic timestamping based on verifiable delay functions*, in: Financial Cryptography and Data Security, Lecture Notes in Comput. Sci. **12059**, Springer, Cham, 2020, pp. 541–558.

[MSW19] M. Mahmoody, C. Smith and D. J. Wu, *A note on the (im)possibility of verifiable delay functions in the random oracle model*, IACR Cryptology ePrint Archive 2019:663, 2019.

[Pie18] K. Pietrzak, *Simple verifiable delay functions*, in: 10th Innovations in Theoretical Computer Science Conference (ITCS 2019), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2019, Art. No. 60, 15 pp.

[RSA78] R. L. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Comm. ACM **21** (1978), 120–126.

[RSS20] L. Rotem, G. Segev and I. Shahaf, *Generic-group delay functions require hidden-order groups*, in: Advances in Cryptology – EUROCRYPT 2020. Part III, Lecture Notes in Comput. Sci. **12107**, Springer, Cham, 2020, pp. 155–180.

[SN77] G. J. Simmons and M. J. Norris, *Preliminary comments on the MIT public-key cryptosystem*, Cryptologia **1** (1977), 406–414.

[ŠNS+16] P. Švenda, M. Nemec, P. Sekan, R. Kvašňovskỳ, D. Formánek, D. Komárek and V. Matyáš, *The million-key question – investigating the origins of RSA public keys*, in: 25th USENIX Security Symposium (USENIX Security 16), USENIX Association, Austin, 2016, pp. 893–910.

[VS04] T. Vasiga and J. Shallit, *On the iteration of certain quadratic maps over $GF(p)$*, Discrete Math. **277** (2004), 219–240.

[War90] R. Warlimont, *Sieving by large prime factors*, Monatsh. Math. **109** (1990), 247–256.

[Wei01] A. Weingartner, *Integers free of prime divisors from an interval, I*, Acta Arith. **98** (2001), 117–131.

[Wes19] B. Wesolowski, *Efficient verifiable delay functions*, in: Advances in Cryptology – EUROCRYPT 2019, Lecture Notes in Comput. Sci. **11478**, Springer, Cham, 2019, pp. 379–407.

## O pretpostavkama niskog reda u RSA grupama

*István András Seres i Péter Burcsi*

Sažetak. U ovom članku pokazujemo da su bitno slabije pretpostavke niskog reda dovoljne za dokazati ispravnost Pietrzakovog protokola za dokaz potenciranja u grupama nepoznatog reda. Ovo čini prvi korak ka boljem razumijevanju asimptotskog računanja složenosti razbijanja ispravnosti protokola. Nadalje, dokazujemo ekvivalentnost (slabije) pretpostavke niskog reda i pretpostavke faktorizacije u RSA grupama za nezanemariv dio modula. Tvrdimo da se u praksi naša redukcija odnosi na znatan broj raspoređenih modula. Naši rezultati imaju kriptografske primjene, od kojih su najvažnije u teoriji nedavno predložene konstrukcije provjerljive funkcije kašnjenja. Na kraju, opisujemo kako certificirati RSA module koji nemaju elemente niskog reda.

István András Seres
Faculty of Informatics, 3in Research Group
ELTE Eötvös Loránd University
1117 Budapest, Hungary
*E-mail*: istvanseres@caesar.elte.hu

Péter Burcsi
Faculty of Informatics, 3in Research Group
ELTE Eötvös Loránd University
1117 Budapest, Hungary
*E-mail*: bupe@inf.elte.hu

*Received*: 27.8.2020.
*Revised*: 30.11.2020.
*Accepted*: 15.12.2020.