Improved Factorization of $N = p^r q^s$

Jean-Sébastien Coron[1] and Rina Zeitoun[2]

[1] University of Luxembourg
jean-sebastien.coron@uni.lu
[2] Oberthur Technologies, Cryptography and Security Group, France
r.zeitoun@oberthur.com

June 1, 2016

Abstract. Boneh *et al.* showed at Crypto 99 that moduli of the form $N = p^r q$ can be factored in polynomial time when $r \geq \log p$. Their algorithm is based on Coppersmith's technique for finding small roots of polynomial equations. Recently, Coron *et al.* showed that $N = p^r q^s$ can also be factored in polynomial time, but under the stronger condition $r \geq \log^3 p$. In this paper, we show that $N = p^r q^s$ can actually be factored in polynomial time when $r \geq \log p$, the same condition as for $N = p^r q$.

1 Introduction

Factoring $N = p^r q$. At Eurocrypt 96, Coppersmith showed how to recover small roots of polynomial equations using lattice reduction [Cop96a,Cop96b]. Coppersmith's technique has found numerous applications in cryptography, in particular the factorization of $N = pq$ when half of the bits of p are known [Cop97].

Coppersmith's technique was later extended to moduli $N = p^r q$ by Boneh, Durfee and Howgrave-Graham (BDH) at Crypto 99 [BDHG99]. They showed that knowing a fraction $1/(r+1)$ of the bits of p is enough for polynomial-time factorization of $N = p^r q$. Therefore when $r \simeq \log p$ only a constant number of bits of p must be known, hence those bits can be recovered by exhaustive search, and factoring $N = p^r q$ becomes polynomial-time [BDHG99]. Such moduli had been suggested by Takagi [Tak98] to significantly speed up RSA decryption; the BDH result shows that Takagi's cryptosystem should not be used with a large r.

Factoring $N = p^r q^s$. In light of the BDH attack, Lim *et al.* in [LKYL00] extended Takagi's cryptosystem to moduli of the form $N = p^r q^s$; namely the generalization to factoring moduli $N = p^r q^s$ was left as an open problem in [BDHG99]. The authors of [LKYL00] obtained an even faster decryption than in Takagi's cryptosystem; in particular, for a 8192-bit RSA modulus $N = p^2 q^3$, decryption becomes 15 times faster than for a standard RSA modulus of the same size.

However, Coron *et al.* have recently described in [CFRZ16] an algorithm to factor $N = p^r q^s$ in deterministic polynomial time when r and/or s is greater than $\log^3 \max(p, q)$. Their method consists in finding a good decomposition of the exponents r and s:

$$\begin{cases} r = u \cdot \alpha + a \\ s = u \cdot \beta + b \end{cases}$$

with large enough integer u, and small enough integers α, β, a, b, so that $N = p^r q^s$ can be rewritten as $N = P^u Q$ where $P = p^\alpha q^\beta$ and $Q = p^a q^b$, and subsequently apply BDH on

$N = P^u Q$ to recover P and Q, and eventually p and q. In BDH the condition for polynomial-time factorization of $N = P^u Q$ is $u = \Omega(\log Q)$. Using lattice reduction and working through tedious arithmetic, the authors show that for any exponent pair (r, s) one can always find integers u, α, β, a and b satisfying $u \simeq r^{2/3}$ and $\alpha, \beta, a, b \simeq r^{1/3}$, which allows them to derive their final condition $r = \Omega(\log^3 \max(p, q))$ for polynomial-time factorization of $N = p^r q^s$.

Our Result. In this paper, we describe an algorithm for factoring moduli of the form $N = p^r q^s$ in polynomial time, under the weaker condition $r = \Omega(\log q)$, the same condition as BDH for $N = p^r q$. Apart from being more efficient than [CFRZ16], our method is also much simpler. Our technique works as follows: since we can assume that $\gcd(r, s) = 1$, from Bézout identity we can find two positive integers α and β such that:

$$\alpha \cdot s - \beta \cdot r = 1$$

This enables to decompose N^α (instead of N previously) as:

$$N^\alpha = (p^r q^s)^\alpha = p^{\alpha r} q^{\alpha s} = p^{\alpha r} q^{\beta r + 1} = \left(p^\alpha q^\beta\right)^r q$$

and apply BDH directly on $N^\alpha = P^r q$ where $P := p^\alpha q^\beta$, and recover p and q. Since for BDH the condition for polynomial-time factorization is $r = \Omega(\log q)$, we obtain exactly the same condition for factoring $N = p^r q^s$. This shows that moduli of the form $N = p^r q^s$ are just as vulnerable as moduli $N = p^r q$ when the exponent r (or s) is large.

2 Background

2.1 Coppersmith's Method

Coppersmith showed in [Cop96b,Cop97] how to find efficiently all small roots of univariate modular polynomial equations. Given a polynomial $f(x)$ of degree δ modulo an integer N of unknown factorization, Coppersmith's method allows to recover in polynomial time in $\log N$ all integers x_0 such that $f(x_0) \equiv 0 \bmod N$ with $|x_0| < N^{1/\delta}$.

A variant of Coppersmith's theorem for univariate modular polynomial equations was obtained by Blömer and May [BM05], using Coppersmith's technique for finding small roots of bivariate integer equations:

Theorem 1 ([BM05, Corollary 14]). *Let N be a composite integer of unknown factorization with divisor $b \geq N^\beta$. Let $f(x) = \sum_i f_i x^i \in \mathbb{Z}[x]$ be a polynomial of degree δ with $\gcd(f_1, \ldots, f_\delta, N) = 1$. Then we can find all points $x_0 \in \mathbb{Z}$ satisfying $f(x_0) = b$ in time polynomial in $\log N$ and δ provided that $|x_0| \leq N^{\beta^2/\delta}$.*

Coppersmith's technique has found many applications in cryptography (see [May10] for a survey), in particular the factorization of $N = pq$ when half of the bits of p are known [Cop97].

2.2 Factoring $N = p^r q$

Coppersmith's technique was later extended to moduli $N = p^r q$ by Boneh, Durfee and Howgrave-Graham (BDH) at Crypto 99 [BDHG99]. They showed that knowing a fraction

$1/(r+1)$ of the bits of p is enough for polynomial-time factorization of $N = p^r q$. Therefore when $r \simeq \log p$ only a constant number of bits of p must be known, hence those bits can be recovered by exhaustive search, and factoring $N = p^r q$ becomes polynomial-time [BDHG99]. We recall their main theorem.

Theorem 2 (BDH). *Let $N = p^r q$ where $q < p^c$ for some c. The factor p can be recovered from N, r, and c by an algorithm with a running time of:*

$$exp\left(\frac{c+1}{r+c} \cdot \log p\right) \cdot \mathcal{O}(\gamma),$$

where γ is the time it takes to run LLL on a lattice of dimension $\mathcal{O}(r^2)$ with entries of size $\mathcal{O}(r \log N)$. The algorithm is deterministic, and runs in polynomial space.

When p and q have similar bitsize we can take $c = 1$; in that case we have $(c+1)/(r+c) = \mathcal{O}(1/r)$ and therefore the algorithm is polynomial time when $r = \Omega(\log p)$. More generally one can take $c = \log q / \log p$, which gives:

$$\frac{c+1}{r+c} \cdot \log p \leq \frac{c+1}{r} \cdot \log p \leq \frac{\frac{\log q}{\log p} + 1}{r} \cdot \log p \leq \frac{\log q + \log p}{r}$$

Therefore a sufficient condition for polynomial-time factorization is $r = \Omega(\log q + \log p)$.

As observed in [CFRZ16], one can actually obtain the simpler condition $r = \Omega(\log q)$, either by slightly modifying the proof of Theorem 2 in [BDHG99], or directly from the Blömer and May variant recalled previously (Theorem 1). We obtain the following theorem. For completeness we provide a proof based on Theorem 1. Note that in the theorem the integer q is prime but p can be any integer.

Theorem 3 (BDH). *Let p and q be two integers with $p \geq 2$ and $q \geq 2$, and q a prime. Let $N = p^r q$. The factors p and q can be recovered in polynomial time in $\log N$ if $r = \Omega(\log q)$.*

Proof. Given $r > 1$ the decomposition $N = p^r q$ is unique for a prime q. One considers the polynomial $f(x) = (P+x)^r$ where P is an integer such that $p = P + x_0$ and the high-order bits of P are the same as the high-order bits of p. Let $b := p^r$ be a divisor of N. The polynomial f satisfies $f(x_0) = (P + x_0)^r = p^r = b$. According to Theorem 1, one can recover x_0 in time polynomial in $\log N$ and r provided that $|x_0| \leqslant N^{\beta^2/r}$, where β is such that $b \geqslant N^\beta$. One can take $b = p^r = N^\beta$, which gives:

$$N^{\beta^2/r} = \left(N^\beta\right)^{\beta/r} = (p^r)^{\beta/r} = p^\beta \quad .$$

Therefore, one gets the condition to recover x_0:

$$|x_0| \ \leqslant \ p^\beta \quad . \tag{1}$$

Moreover from $p^r = N^\beta = (p^r q)^\beta$ we get:

$$\beta \ = \ \frac{r \log p}{r \log p + \log q} \ = \ \frac{1}{1 + \frac{\log q}{r \log p}} \ \geqslant \ 1 - \frac{\log q}{r \log p} \quad .$$

Therefore we have:

$$p^{\beta} \geqslant p^{1-\frac{\log q}{r\log p}} = p \cdot \left(p^{\frac{\log q}{\log p}}\right)^{-1/r} = p \cdot q^{-1/r} \quad . \tag{2}$$

By combining inequalities (1) and (2), one gets the following sufficient condition:

$$|x_0| \leqslant p \cdot q^{-1/r} \quad .$$

Therefore it suffices to perform exhaustive search on $q^{1/r}$ possible values for the high-order bits of p. When $r = \Omega(\log q)$ we have $q^{1/r} = \mathcal{O}(1)$, and therefore one can recover p and q in time polynomial in $\log N$. □

3 Improved Factorization of $N = p^r q^s$

We show that moduli of the form $N = p^r q^s$ can be factored in polynomial time under the condition $r = \Omega(\log q)$; this improves [CFRZ16] which required $r = \Omega(\log^3 \max(p, q))$; our technique is also much simpler. We can assume that $r > s$, since otherwise we can swap p and q. We can also assume that $\gcd(r, s) = 1$, since otherwise one should consider $N' = N^{1/\gcd(r,s)}$. Furthermore, we assume that the exponents r and s are known; otherwise they can be recovered by exhaustive search in time $\mathcal{O}(\log^2 N)$.

Theorem 4. *Let $N = p^r q^s$ be an integer of unknown factorization with $\gcd(r, s) = 1$. Given N as input, one can recover the prime factors p and q in polynomial time in $\log N$ under the condition $r = \Omega(\log q)$.*

Proof. Since $\gcd(r, s) = 1$, from Bézout's identity there exist two positive integers α and β such that:

$$\alpha \cdot s - \beta \cdot r = 1 \ ,$$

where we can take $0 < \alpha < r$ since $\alpha \equiv s^{-1} \pmod r$. Therefore we can write:

$$N^{\alpha} = (p^r q^s)^{\alpha} = p^{\alpha r} q^{\alpha s} = p^{\alpha r} q^{\beta r + 1} = \left(p^{\alpha} q^{\beta}\right)^r q$$

Therefore letting $P := p^{\alpha} q^{\beta}$, we obtain $N^{\alpha} = P^r q$. One can thus apply Theorem 3 on N^{α}, which enables to recover the integers P and q from $N^{\alpha} = P^r q$ in polynomial time in $\log(N^{\alpha})$, under the condition $r = \Omega(\log q)$. Since $\alpha < r < \log N$, this enables to recover the factorization of N in time polynomial in $\log N$ under that condition. □

References

[BDHG99] Dan Boneh, Glenn Durfee, and Nick Howgrave-Graham. Factoring $n = p^r q$ for large r. In *Advances in Cryptology - Proc. CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 326–337. Springer, 1999.

[BM05] Johannes Blömer and Alexander May. A tool kit for finding small roots of bivariate polynomials over the integers. In *Advances in Cryptology - Proc. EUROCRYPT '05*, volume 3494 of *Lecture Notes in Computer Science*, pages 251–267. Springer, 2005.

[CFRZ16] Jean-Sébastien Coron, Jean-Charles Faugère, Guénaël Renault, and Rina Zeitoun. Factoring $N = p^r q^s$ for large r and s. In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, Proceedings*, pages 448–464, 2016.

[Cop96a] Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In *Advances in Cryptology - Proc. EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 1996.

[Cop96b] Don Coppersmith. Finding a small root of a univariate modular equation. In *Advances in Cryptology - Proc. EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 155–165. Springer, 1996.

[Cop97] Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997. Journal version of [Cop96b,Cop96a].

[LKYL00] Seongan Lim, Seungjoo Kim, Ikkwon Yie, and Hongsub Lee. A generalized Takagi-cryptosystem with a modulus of the form $p^r q^s$. In *Progress in Cryptology - INDOCRYPT 2000, First International Conference in Cryptology in India, Calcutta, India, December 10-13, 2000, Proceedings*, pages 283–294, 2000.

[May10] Alexander May. Using LLL-reduction for solving RSA and factorization problems. In *The LLL Algorithm - Survey and Applications*, pages 315–348. 2010.

[Tak98] Tsuyoshi Takagi. Fast RSA-type cryptosystem modulo $p^k q$. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 318–326, 1998.