Implicit Factoring with Shared Most Significant and Middle Bits

Jean-Charles Faugère, Raphaël Marinier, and Guénaël Renault

UPMC, Université Paris 06, LIP6
INRIA, Centre Paris-Rocquencourt, SALSA Project-team
CNRS, UMR 7606, LIP6
4, place Jussieu
75252 Paris, Cedex 5, France
jean-charles.faugere@inria.fr, raphael.marinier@polytechnique.edu,
guenael.renault@lip6.fr

Keywords: implicit factorization, lattices, RSA

The corresponding paper version of this extended abstract is accepted for PKC2010 [3]

The problem of factoring integers given additional information about their factors has been studied since 1985. In [6], Rivest and Shamir showed that $N = pq$ of bit-size n and with balanced factors ($\log_2(p) \approx \log_2(q) \approx \frac{n}{2}$) can be factored in polynomial time as soon as we have access to an *oracle* that returns the $\frac{n}{3}$ most significant bits (MSBs) of p. Beyond its theoretical interest, the motivation behind this is mostly of cryptographic nature. In fact, during an attack of an RSA-encrypted exchange, the cryptanalyst may have access to additional information beyond the RSA public parameters (e,N), that may be gained for instance through side-channel attacks revealing some of the bits of the secret factors. Besides, some variations of the RSA Cryptosystem purposely leak some of the secret bits (for instance, [8]). In 1996, Rivest and Shamir's results were improved in [2] by Coppersmith applying lattice-based methods to the problem of finding small integer roots of bivariate integer polynomials (the now so-called *Coppersmith's method*). It requires only half of the most significant bits of p to be known to the cryptanalyst (that is $\frac{n}{4}$).

In PKC 2009, May and Ritzenhofen [5] significantly reduced the power of the oracle. Given an RSA modulus $N_1 = p_1q_1$, they allow the oracle to output a new and different RSA modulus $N_2 = p_2q_2$ such that p_1 and p_2 share at least t least significant bits (LSBs). Note that the additional information here is only *implicit*: the attacker does not know the actual value of the t least significant bits of the p_i's, he only knows that p_1 and p_2 share them. In the rest of the paper, we will refer to this problem as the problem of *implicit factoring*. When q_1 and q_2 are α-bit primes, May and Ritzenhofen's lattice-based method rigorously finds in quadratic time the factorization of N_1 and N_2 when $t \geq 2\alpha + 3$. Besides, their technique heuristically generalizes to $k-1$ oracle queries that give access to k different RSA moduli $N_i = p_iq_i$ with all the p_i's sharing t least significant bits. With $k-1$ queries the bound on t improves to: $t \geq \frac{k}{k-1}\alpha$. Note that these results are of interest for unbalanced RSA moduli: for instance, if $N_1 = p_1q_1$, $N_2 = p_2q_2$ are 1000-bit RSA moduli and the q_i's are 200-bit primes, knowing that p_1 and p_2 share at least 403 least significant bits out of 800 is enough to factorize N_1 and N_2 in polynomial time. Note also that the method absolutely requires that the shared bits be the least significant ones. They finally apply their method to factorize k n-bit balanced RSA moduli $N_i = p_iq_i$ under some conditions and with an additional exhaustive search of $2^{\frac{n}{4}}$.

Very recently, in [7], Sarkar and Maitra applied Coppersmith and Gröbner-basis techniques on the problem of implicit factoring, and improved heuristically the bounds in some of the cases. Contrary to

[5], their method applies when either (or both) LSBs or MSBs of p_1, p_2 are shared (or when bits in the middle are shared). Namely, in the case of shared LSBs they obtain better theoretical bounds on t than [5] as soon as $\alpha \geq 0.266n$. Besides, their experiments often perform better than their theoretical bounds, and they improve in practice the bound on t of [5] when $\alpha \geq 0.21n$. Note finally that their bounds are very similar in the two cases of shared MSBs and shared LSBs. Readers interested in getting their precise bounds may refer to their paper [7].

Unfortunately, Sarkar and Maitra's method is heuristic even in the case of two RSA moduli, and does not generalize to $k \geq 3$ RSA moduli. In fact, when the p_i's share MSBs and/or LSBs, their method consists in building a polynomial f_1 in three variables, whose roots are $(q_2+1, q_1, \frac{p_1-p_2}{2^\gamma})$, where γ is the number of shared LSBs between p_1 and p_2. That is, $\frac{p_1-p_2}{2^\gamma}$ represents the part of $p_1 - p_2$ where the shared bits do not cancel out. To find the integer roots of f_1, they use the Coppersmith-like technique of [4] which consists in computing two (or more) new polynomials $f_2, f_3, \ldots$ sharing the same roots as f_1. If the variety defined by $f_1, f_2, f_3, \ldots$ is 0-dimensional, then the roots can be easily recovered computing resultants or Gröbner basis. However, with an input polynomial with more than two variables, the method is heuristic: there is no guarantee for the polynomials $f_1, f_2, f_3, \ldots$ to define a 0-dimensional variety. We reproduced the results of Sarkar and Maitra and we observed that $f_1, f_2, f_3, \ldots$ almost never defined a 0-dimensional variety. They observed however that it was possible to recover the roots of the polynomials directly by looking at the coefficients of the polynomials in the Gröbner basis of the ideal generated by the f_i's, even when the ideal was of positive dimension. The assumption on which their work relies is that it will always be possible. For instance, in the case of shared MSBs between p_1 and p_2, they found in their experiments that the Gröbner basis contained a polynomial multiple of $x - \frac{q_2}{q_1}y - 1$ whose coefficients lead immediately to the factorization of N_1 and N_2. They support their assumption by experimental data: in most cases their experiments perform better than their theoretical bounds. It seems nevertheless that their assumption is not fully understood.

Our contribution consists of a novel and rigorous lattice-based method that address the implicit factoring problem when p_1 and p_2 share *most* significant bits. That is, we obtained an analog of May and Ritzenhofen's results for shared MSBs, and our method is rigorous contrary to the work of Sarkar and Maitra in [7]. Namely, let $N_1 = p_1q_1$ and $N_2 = p_2q_2$ be two RSA moduli of same bit-size n. If q_1, q_2 are α-bit primes and p_1, p_2 share t most significant bits, our method provably factorizes N_1 and N_2 as soon as $t \geq 2\alpha + 3$ (which is the same as the bound on t for least significant bits in [5]). This is the first rigorous bound on t when p_1 and p_2 share most significant bits. From this method, we deduce a new heuristic lattice-based for the case when p_1 and p_2 share t bits in the middle. Moreover, contrary to [7], these methods heuristically generalize to an arbitrary number k of RSA moduli and do not depend on the position of the shared bits in the middle, allowing us to factorize k RSA moduli as soon as $t \geq \frac{k}{k-1}\alpha + 6$ (resp. $t \geq \frac{2k}{k-1}\alpha + 7$) most significant bits (resp. bits in the middle) are shared between the p_i's (more precise bounds are stated later in this paper). A summary of the comparison of our method with the methods in [5] and [7] can be found in table 1.

Let's give the main idea of our method with 2 RSA moduli in the case of shared MSB's. Consider the lattice L spanned by the row vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ of the following matrix:

$$\begin{pmatrix} K & 0 & N_2 \\ 0 & K & -N_1 \end{pmatrix} \quad \text{where } K = \lfloor 2^{n-t+\frac{1}{2}} \rfloor$$

Consider also the following vector in L:

$$\mathbf{v_0} = q_1\mathbf{v_1} + q_2\mathbf{v_2} = (q_1K, q_2K, q_1q_2(p_2 - p_1))$$

The key observation is that the t shared significant bits of p_1 and p_2 cancel out in the algebraic relation $q_1N_2 - q_2N_1 = q_1q_2(p_2 - p_1)$. Furthermore, we choose K in order to force the coefficients of a shortest

Table 1: Comparison of our results against the results of [5] and [7]

k (number of RSA moduli)	May, Ritzenhofen's Results [5]	Sarkar, Maitra's Results [7]	Our results
$k = 2$	When p_1, p_2 share t LSBs: rigorous bound of $t \geq 2\alpha + 3$ using 2-dimensional lattices of $\mathbb{Z}^2$.	When p_1, p_2 share either t LSBs or MSBs: heuristic bound better than $t \geq 2\alpha + 3$ when $\alpha \geq 0.266n$, and experimentally better when $\alpha \geq 0.21n$. In the case of t shared bits in the middle, better bound than $t \geq 4\alpha + 7$ but depending on the position of the shared bits. Using 46-dimensional lattices of $\mathbb{Z}^{46}$	When p_1, p_2 share t MSBs: rigorous bound of $t \geq 2\alpha + 3$ using 2-dimensional lattices of $\mathbb{Z}^3$. In the case of t bits shared in the middle: heuristic bound of $t \geq 4\alpha + 7$ using 3-dimensional lattices of $\mathbb{Z}^3$.
$k \geq 3$	When the p_i's all share t LSBs: heuristic bound of $t \geq \frac{k}{k-1}\alpha$ using k-dimensional lattices of $\mathbb{Z}^k$.	Cannot be directly applied.	When the p_i's all share t MSBs (resp. bits in the middle): heuristic bound of $t \geq \frac{k}{k-1}\alpha + \delta_k$ (resp. $t \geq \frac{2k}{k-1}\alpha + \delta_k$), with $\delta_k \leq 6$ (resp. ≤ 7) and using k-dimensional ($\frac{k(k+1)}{2}$-dimensional) lattices of $\mathbb{Z}^{\frac{k(k+1)}{2}}$.

vector of L on the basis $(\mathbf{v_1}, \mathbf{v_2})$ to be of the order of $2^\alpha \approx q_1 \approx q_2$. We proved a result stating that $\mathbf{v_0}$ is indeed a shortest vector of L (thus N_1 and N_2 can be factored in polynomial time) as soon as $t \geq 2\alpha + 3$. Besides, we generalized this construction to an arbitrary number of k RSA moduli such that a small vector of the lattice harnesses the same algebraic relation, and to shared middle bits. However, the generalized constructions in both cases become heuristic: we use the Gaussian heuristic to find a condition on t for this vector to be a shortest of the lattice. More precisely, we obtained the following results

Theorem 1. *Let $N_1 = p_1q_1, N_2 = p_2q_2$ be two n-bit RSA moduli, where the q_i's are α-bit primes and the p_i's are primes that share t most significant bits. If $t \geq 2\alpha + 3$, then N_1 and N_2 can be factored in quadratic time in n.*

Let $\mathscr{C}(k, s, B)$ be the time to find a shortest vector of a k-dimensional lattice of $\mathbb{Z}^s$ given by B-bit basis vectors. We have the following generalization and application which are **stated under Gaussian heuristic** (we assume that if $\pm\mathbf{v_0}$ is a vector of a d-dimensional lattice L with norm smaller than $\sqrt{\frac{d}{2\pi e}}\,\mathrm{Vol}(L)^{\frac{1}{d}}$ then it is a shortest vector of L):

Theorem 2. *Let $N_1 = p_1q_1, \ldots, N_k = p_kq_k$ be k n-bit RSA moduli, with the q_i's being α-bit primes, and the p_i's being primes that all share t most significant bits. The N_i's can be factored in time $\mathscr{C}(k, \frac{k(k+1)}{2}, n)$, as soon as*

$$t \geq \frac{k}{k-1}\alpha + 1 + \frac{k}{2(k-1)}\left(2 + \frac{\log_2(k)}{k} + \log_2(\pi e)\right)$$

Theorem 3. *Let $N_1 = p_1q_1, \ldots, N_k = p_kq_k$ be k n-bit RSA moduli, where the q_i's are α-bit primes and the p_i's are primes that all share t bits from the position t_1 to $t_2 = t_1 + t$. The N_i's can be factored in time $\mathscr{C}(\frac{k(k+1)}{2}, \frac{k(k+1)}{2}, n)$, as soon as*

$$t \geq 2\alpha + \frac{2}{k-1}\alpha + \frac{k+1}{2(k-1)}\log_2(2\pi e)$$

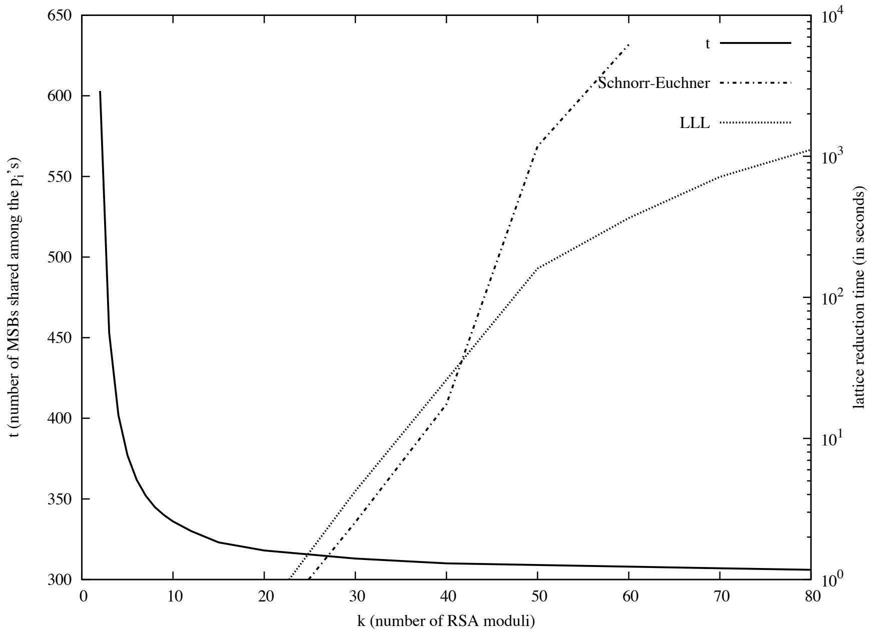
We support these results by experimental facts. In order to check the validity of Gaussian heuristic in our case and the quality of our bounds on t, we implemented the methods on Magma 2.15 [1].
The MSB case. We generated many random 1024-bit RSA moduli, for various values of α and t. We observed that the results were similar for other values of n. In the case where $k = 2$, we used the Lagrange reduction to find with certainty a shortest vector of the lattice, and for $3 \leq k \leq 40$ we compared Schnorr-Euchner's algorithm (that provably outputs a shortest vector of the lattice) with LLL (that gives an exponential approximation of a shortest vector). We used only LLL for $k = 80$.
We conducted experiments for $k = 2, 3, 10, 40$ and 80, and for several values for α. In the rigorous case $k = 2$, we observed that the attack consistently goes one bit further with 100% success rate than our bound in Theorem 1. In all our experiments concerning the heuristic cases $k \geq 3$, we observed that we had 100% success rate (thus, Gaussian heuristic was always true in our case) when t was within the bound of Theorem 2. That means that the assumption concerning the Gaussion heuristic was always true in our experiments. Moreover, we were often able to go a few bits (up to 3) beyond the theoretical bound on t. When the success rate was not 100% (that is, beyond our experimental bounds on t), we found that Gaussian heuristic was not true in a very limited number of the cases (less than 3%). Finally, up to dimension 80, LLL was always sufficient to find $\mathbf{v_0}$ when t was within the bound of Theorem 2, and Schnorr-Euchner's algorithm allowed us to go one bit further than LLL in dimension 40.
The middle bits case. Contrary to the case of shared MSBs, Gaussian heuristic may fail when we apply our method with shared bits in the middle since there may exist some exceptional shortest vectors which does not correspond to the solution of our problem. When $k = 2$ the phenomenon of exceptional short vectors rarely appeared when t was within the bound of Theorem 3 (less than 1% of failure and did not depend on the position of the bits, moreover, we were generally allowed to go 2 or 3 bits further with 90% of success). When $k \geq 3$ it was not still the case. When Schnorr-Euchner's algorithm did not return $\mathbf{v_0}$, we tried to find it in a reduced basis computed by LLL. Our experiments showed that for the same size of problems the rate of success is approximately 80% when t was within the bound of Theorem 3 and allowed us to go one or two bits further with success rate ≈50%.
Efficiency comparisons. Additionally, we show in Table 2 the lowest value of t with 100% success rate and the running-time of LLL and Schnorr-Euchner's algorithm for several values of k (k RSA moduli with p_i's factors sharing t MSBs). For each k, we show the worst running-time we encountered when running 10 tests on an Intel Xeon E5420 at 2.5Ghz. We see that all individual tests completed in less than 1 second for $2 \leq k \leq 20$. We used Schnorr-Euchner's algorithm up to $k = 60$ where it took at most 6200 seconds. LLL completes under one minute for $20 \leq k \leq 40$ and in less than 30 minutes for $40 \leq k \leq 80$.

Applications of implicit factoring have not yet been extensively studied, and we believe that they will develop. The introduction of [5] gives some ideas for possible applications. They include destructive applications with malicious manipulation of public key generators, as well as possibly constructive ones. Indeed, our work shows that when $t \geq 2\alpha + 3$, it is as hard to factorize $N_1 = p_1q_1$, as generating $N_2 = p_2q_2$ with p_2 sharing t most significant bits with p_1. This problem could form the basis of a cryptographic primitive.

References

1. Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system I: The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).
2. Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In Ueli M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 1996.

Table 2: Running time of LLL and Schnorr-Euchner's algorithm, and bound on t as k grows. (Shared MSBs with $\alpha = 300$ and $n = 1024$)



3. Jean-Charles Faugère, Raphaël Marinier, and Guénaël Renault. Implicit factoring with shared most significant and middle bits. In P.Q. Nguyen and D. Poincheval, editors, *PKC*, volume 6056 of *Lecture Notes in Computer Science*, pages 70–87. Springer-Verlag, 2010.
4. Ellen Jochemsz and Alexander May. A strategy for finding roots of multivariate polynomials with new applications in attacking rsa variants. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 267–282. Springer, 2006.
5. Alexander May and Maike Ritzenhofen. Implicit factoring: On polynomial time factoring given only an implicit hint. In Stanislaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2009.
6. Ronald L. Rivest and Adi Shamir. Efficient factoring based on partial information. In Franz Pichler, editor, *EUROCRYPT*, volume 219 of *Lecture Notes in Computer Science*, pages 31–34. Springer, 1985.
7. Santanu Sarkar and Subhamoy Maitra. Further Results on Implicit Factoring in Polynomial Time. *Advances in Mathematics of Communications*, 3(2):205–217, 2009.
8. Scott A. Vanstone and Robert J. Zuccherato. Short rsa keys and their generation. *J. Cryptology*, 8(2):101–114, 1995.