New RSA Vulnerabilities Using Lattice Reduction Methods

Dissertation Thesis

by

Alexander May

October 19, 2003

Reviewer: Prof. Dr. Johannes Blömer (University of Paderborn)

Prof. Dr. Joachim von zur Gathen (University of Paderborn)

Prof. Dr. Claus-Peter Schnorr (University of Frankfurt/Main)

"Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve."
Edgar Allan Poe

Acknowledgments:

First of all, I want to thank my advisor Johannes Blömer for his support in all stages of this thesis.

I had the chance to meet many other researchers in the cryptographic community who also influenced this work. Especially, I want to thank Nick Howgrave-Graham, Phong Nguyen, Jean-Pierre Seifert, Joe Silverman and Damien Stehlé.

Furthermore, I want to thank the International Association for Cryptologic Research (IACR) that offered me stipends for the conferences CRYPTO 2002 and CRYPTO 2003.

For proof reading and many useful comments, I want to thank Birgitta Grimm, Martin Otto, Harald Räcke, Kay Salzwedel, Christian Sohler and Manuela Thygs.

Further, I want to thank Friedhelm Meyer auf der Heide and Emo Welzl for giving me the opportunity to join their research groups.

Enjoy the thesis!

Contents

1 Introduction

> *"The mathematician's patterns, like the painter's or the poet's, must be beautiful; the ideas, like the colors or the words must fit together in a harmonious way. Beauty is the first test; there is no permanent place in this world for ugly mathematics."*
> G.H. Hardy (1877-1947)

Public Key Cryptography – A concept straight from the Book

In 1976, Whitfield Diffie and Martin Hellman [26] came up with a new idea that lead to a revolution in the field of cryptography: They designed a protocol, in which two people can agree on a common secret key over an insecure channel like the internet. This so-called key exchange protocol is named the Diffie-Hellman key exchange. The basic principle in their protocol is the use of a so-called *one-way function*. Informally, this is a mathematical function that is easy to compute but computationally infeasible to invert.

As a candidate for such an one-way function, Diffie and Hellman proposed the *discrete exponentiation function*. Let n be an integer and denote by $\mathbb{Z}_n := \{0, 1, \ldots, n-1\}$ the ring of integers modulo n. Furthermore, let p be a prime. It is not hard to see that $\mathbb{Z}_p$ together with the multiplication forms an abelian, cyclic group G of order $p-1$. Let $g \in G$ be a generator of G. Then the function $f(g, x) = g^x \bmod p$ is called the *discrete exponentiation function*.

Assume now that two persons, Alice and Bob, want to commit to a secret key k. In the Diffie-Hellman protocol, Alice chooses a secret number $a \in \mathbb{Z}_{p-1}$, applies the discrete exponentiation function $f(g, a) = g^a \bmod p$ and sends the value $g^a \bmod p$ to Bob. Likewise, Bob chooses a secret number $b \in \mathbb{Z}_{p-1}$ and sends the value of $f(g, b) = g^b \bmod p$ to Alice. Now, Alice can compute the value $k = f(g^b, a) = g^{ab} \bmod p$ and Bob can compute the same value $k = f(g^a, b) = g^{ab} \bmod p$.

On the other hand, an eavesdropper, Eve, who listens to Alice's and Bob's communication only learns the values g^a and g^b. The problem of computing the value g^{ab} on input (g, g^a, g^b) is known as the Diffie-Hellman problem. Obviously, one can reduce the Diffie-Hellman problem to the computation of the inverse of the discrete exponentiation function: On input (g, g^x), compute the value of x. This inversion problem is also called the *discrete logarithm problem.* If Eve could solve the discrete logarithm problem efficiently, then she could compute Alice's secret number a and $k = f(g^b, a)$.

However, it is not known in general whether Eve really has to solve the discrete logarithm problem in order to solve the Diffie-Hellman problem. In other words, it is unknown whether the Diffie-Hellman problem is *polynomial time equivalent* to the discrete logarithm problem. Polynomial time equivalence of both problems means that every algorithm that solves the Diffie-Hellman problem in polynomial time can be turned into an algorithm that solves the discrete logarithm problem in polynomial time and vice versa. For some special kinds of groups G, the polynomial time equivalence of both problems has been shown by Maurer and Wolf [47].

The discovery of Diffie and Hellman was the foundation of a new field in cryptography: the *public key cryptography.* With the Diffie-Hellman protocol, people are able to securely exchange a secret key k over an insecure channel that may be eavesdropped. After the key exchange, they can use the key k in a so-called symmetric key cryptosystem to communicate to each other. A cryptosystem is called symmetric if the same key is used for both: encryption and decryption.

However, there is a drawback in this method: Assume that we have n users in an insecure network and we want to allow each user to communicate securely to each other. Then, we have to run the Diffie-Hellman protocol $\binom{n}{2}$ times to establish a secret key for each pair of users. Even more, every user has to store $n-1$ secret keys. Clearly, for networks like the internet with millions of users this is not practical.

RSA – An asymmetric cryptosystem

In 1978, Ron Rivest, Adi Shamir and Leonard Adleman [56] came up with a very elegant solution for the key exchange problem: They invented the first *public key cryptosystem*, called the RSA scheme. The fundamental improvement compared with Diffie-Hellman is that every user needs only one pair of keys in order to communicate to each other in the network: the public key/secret key pair.

The new idea in the RSA scheme is that different keys are used for encryption and decryption of a message: The public key is used for encrypting messages whereas the secret key is used for the decryption. Therefore, RSA is also called an *asymmetric cryptosystem.* Actually, the idea of asymmetric cryptography was already due to Whitfield Diffie but Rivest, Shamir and Adleman were the first who designed an asymmetric cryptosystem [1]. The idea of an asymmetric scheme is as follows:

Assume that Alice has a public key k_p and a secret key k_s. She publishes k_p on the network, such that everyone can access her public key. Now, if Bob wants to send a

[1]In the late 90s, it became public knowledge that the principle of asymmetric cryptography had been first mentioned 1969 in a work by James Ellis who worked for the GCHQ, a top-secret English agency. In 1973, Clifford Cocks from GCHQ developed a cryptosystem which is completely analogous to the RSA scheme. One year later, Malcolm Williamson (also from GCHQ) discovered a key exchange protocol similar to the Diffie-Hellman protocol. But due to the agency's secrecy policy the reseachers could not publish their results.

message m to Alice, he encrypts m with Alice's public key k_p. In order to recover m from the ciphertext, Alice uses her secret key k_s in the decryption process.

In the case of the RSA scheme, the idea of an asymmetric scheme is realized in the following way. Alice chooses two large primes p and q and computes the product $N = pq$. Then she selects a pair of integers (e, d), $e \cdot d > 1$ which satisfies a certain relation:

$$m^{ed} = m \bmod N \quad \text{for all messages } m \in \mathbb{Z}_N,$$

where $\mathbb{Z}_N$ denotes the ring of integers modulo N. In fact, $ed = 1 \bmod \phi(N)$ where $\phi(N) := (p-1)(q-1)$ is Euler's totient function. Alice is able to compute a pair (e, d) satisfying such a relation since she knows the prime factorization of N. She publishes the tuple (N, e) which serves as her public key k_p and she keeps the key $k_s = d$ secret.

Now anyone who wants to send a message $m \in \mathbb{Z}_N$ to Alice takes her public key (N, e) and computes the ciphertext $m^e \bmod N$. Similar to the Diffie-Hellman protocol, the so-called *RSA encryption function* $f(x) = x^e \bmod N$ is assumed to be a one-way function. Hence an attacker who knows the ciphertext $m^e \bmod N$ together with the public key (N, e) should not be able to compute the value of m.

But in contrast to the Diffie-Hellman protocol, Alice wants to invert the RSA encryption function in order to read the message m, i.e., she wants to compute the e^{th} root of m^e modulo N. She can decrypt by using her secret key $k_s = d$, namely she simply computes $(m^e)^d = m \bmod N$. That means Alice possesses a trapdoor that allows her to invert the encryption function: the secret key d or likewise the factorization of N from which d can be derived. Therefore, we also call the RSA encryption function a *trapdoor one-way function*.

Unfortunately, it is not known whether the discrete exponentiation function or the RSA encryption function are indeed one-way functions. Even worse, one does not even know whether one-way functions exist at all. On the other hand, in the standard computational model there are no algorithms known that invert the discrete exponentiation function or the RSA encryption function in polynomial time. Hence, these functions are often referred to as *candidate one-way functions*. We want to remark that an algorithm of Shor [61] solves both problems on a quantum computer in polynomial time, but it seems to be a very hard task to build quantum computers in practice.

In the RSA scheme, we have an analogy to the relation between the Diffie-Hellman problem and the discrete logarithm problem: Inverting the RSA encryption function can be reduced to the problem of factoring N. However, it is not known whether these two problems are polynomial time equivalent. In the literature it is often (misleadingly) stated that the security of RSA is based on the hardness of factoring. This means that RSA cannot be secure if factoring is feasible. On the other hand, one should be aware that RSA must not be secure if factoring is computationally infeasible: There may be other ways to compute e^{th} roots modulo N, i.e., other ways to invert the RSA encryption function without factoring.

In todays public key cryptography, the security of the theoretically and practically most important systems relies either on the hardness of inverting the discrete exponentiation function or on the hardness of inverting the RSA encryption function (respectively the hardness of factoring). Unfortunately, public key cryptosystems that are based on the discrete exponentiation function like the ElGamal scheme [29] and the Cramer-Shoup scheme [24] suffer from the fact that they have message expansion rate of at least two. That means, the ciphertext is at least twice as long as the underlying plaintext. This might be the main reason that these schemes are used far less in practice than RSA.

In this thesis we will concentrate on the RSA encryption function. The RSA scheme is the most popular and best studied public key cryptosystem. Since its invention 25 years ago, it has received a lot of attention in both academia and industry. The RSA scheme is implemented as a standard in every web browser and it is commonly used to secure financial transactions. The big impact of the RSA scheme on many real-life applications is one of the reasons that Rivest, Shamir and Adleman received in 2003 the famous Turing Award for their invention of the scheme.

Due to its importance and its beautiful, simple structure the RSA scheme has also attracted many cryptanalysts (for a survey see [9]). But despite intensive research efforts, from a mathematical point of view the only known method to break the RSA scheme is the most obvious one: Find the factorization of N. There are numerous other so-called side channel attacks on the scheme, but these attacks do not directly try to break the scheme itself. Instead, an attacker in a side-channel attacks tries to learn information about the secret key d by attacking physical implementations of RSA.

> *"The source of all great mathematics is the special case, the concrete example. It is frequent in mathematics that every instance of a concept of seemingly great generality is in essence the same as a small and concrete special case."*
> Paul R. Halmos

Our Goal

In this thesis we try to invert the RSA encryption function by solving the factorization problem. Our goal is to design algorithms that factor an RSA modulus $N = pq$ in time polynomial in the bit-size of N. Unfortunately, as stated before, it is not known whether there exists a polynomial time algorithm that factors a composite integer N. The asymptotically best algorithm, the Number Field Sieve, has running time subexponential in the bit-size of N. On the other hand, this does not imply that there are no practically interesting special cases for which the factoring problem turns out to be efficiently solvable. Therefore, we try to relax the factorization problem in order to obtain polynomial time algorithms.

Goal: Identify special instances of the factorization problem that can be solved in polynomial time.

There are many interesting scenarios in practice, where an attacker might be able to get some additional information about the factorization of N from the parameters that are used in the RSA scheme. One can view this additional information as a hint how to find the factorization of N. In this thesis, we mainly deal with two sorts of hints in the RSA parameters that turn our factorization problem into an efficiently solvable problem:

Information encoded in the public exponent e: The main drawback of RSA is its efficiency. Therefore, it is tempting for crypto-designers to speed up the RSA encryption/decryption process by choosing key tuples (e, d) of a special structure. But then, the public exponent e may contain useful information how to factor the modulus N.

Partial knowledge of the secret exponent d: In many side-channel attacks on RSA, an attacker gets knowledge of the bits of d. We study the following question: How many bits of d are sufficient in order to find the factorization of N.

All of the factorization algorithms that we design in this thesis will run on input N and the additional information of our hint and will output the prime factors p, q in time polynomial in the bit-size of N. The main method that we use in order to design our algorithms is an elegant approach proposed by Coppersmith [20] for finding small roots of modular polynomial equations. This approach in turn is based on the famous L^3-algorithm by Lenstra, Lenstra and Lovász [44] for finding short lattice vectors.

An Overview of our results

We briefly describe the organization of the thesis and present the main results.

Chapter 2: In this chapter, we define the notion of a public key cryptosystem. We present the RSA scheme and discuss its efficiency and security. Furthermore, we define some basics of lattice theory.

Chapter 3: We introduce Coppersmith's method [20] for finding small roots of modular polynomial equations. We slightly generalize Coppersmith's theorem for the univariate polynomial case. To our knowledge, this new formulation of Coppersmith's method covers all applications of the univariate case given in the literature. We present some of the most important applications:

- Attacking RSA with small e by knowing parts of the message.
- Factoring $N = pq$ by knowing half of the bits of p.
- Factoring $N = p^r q$ for large r.

Finally, we explain how Coppersmith's method can be extended (heuristically) to multivariate polynomial equations.
(Parts of the results in this chapter were published at PKC 2004 [49]*.)*

Chapter 4: We introduce a generalization of Wiener's famous attack [71] on the RSA scheme. In 1990, Wiener observed that information encoded in the public exponent e may help to factor N. Namely, he showed that every e that corresponds to a secret exponent $d \leq N^{\frac{1}{4}}$ yields the factorization of N in polynomial time.

We extend Wiener's result by showing that every e which corresponds to some $d = \frac{d_1}{d_2} \bmod \phi(N)$ with

$$d_1 \leq N^{\frac{1}{4}} \quad \text{and} \quad |d_2| \leq N^{\frac{1}{4}} d_1$$

yields the factorization of N. That means, not only small secret exponents d lead to the factorization of N but also exponents d that have a "small decomposition" in d_1 and d_2. As an application of our new attack, we cryptanalyze an RSA-type cryptosystem that was proposed in 2001 by Yen, Kim, Lim and Moon [73, 74].

Furthermore, we introduce the notion of weak public keys. Informally, a key (N, e) is weak if it yields the factorization of N in polynomial time. We prove that our new attack indentifies $\Omega(N^{\frac{3}{4}-\epsilon})$ weak RSA keys (N, e).
(Parts of the results in this chapter were published at PKC 2004 in a joint work with Johannes Blömer [6]*.)*

Chapter 5: In this chapter, we address an open problem that was raised in the work of Wiener [71]: Is it possible to factor N if not d itself is small but if the value $d_p := d \bmod p - 1$ is small? The value d_p is often used in fast variants of the RSA decryption process. It is tempting to use small values of d_p in order to speed up the decryption process even further.

We derive attacks for small d_p but unfortunately our attacks are restricted to the case of unbalanced prime factors p and q. In our scenario, the bit-size of the prime factor q has to be significantly smaller than the bit-size of p.

Our first method works whenever $q \leq N^{0.382}$. Our second approach works for prime factors $q \leq N^{\frac{3}{8}}$. This bound is slightly worse than the bound of the first method, but for small q the second approach yields better results: It allows an attacker to factor N for larger values of d_p.
(Parts of the results in this chapter were published at Crypto 2002 [48]*.)*

Chapter 6: We present new partial key exposure attacks on the RSA scheme. Partial key exposure attacks on RSA are attacks where an adversary knows a fraction of the bits of d — for instance he may use side-channel attacks to learn some bits of d. These partial key exposure attacks were introduced in a work of Boneh, Durfee and Frankel [14] in 1999, where the authors raised the question whether there exist attacks on RSA beyond the bound $e = O(\sqrt{N})$.

We answer this question in the two most interesting cases, where an attacker knows either the most significant bits (MSBs) of d or the least significant bits (LSBs) of d.

For the MSBs, we present an attack that works up to the bound $e < N^{0.725}$. In the case of LSBs, our new attack even works up to $e < N^{\frac{7}{8}}$.

Furthermore, we design partial key exposure attacks when an attacker knows bits of the value $d_p := d \bmod p-1$ (or symmetrically of $d_q := d \bmod q-1$). As noticed before, d_p is often used in fast variants of the decryption process. Our results are strong when the public exponent is small: We need only half of the bits of d_p (either MSBs or LSBs) in order to factor N. Since normally the bit-size of p is half of the bit-size of N, this is only an amount of a quarter of the bits of N.

Furthermore, we extend our attacks to moduli of the form $N = p^r q$ for some $r > 1$. These moduli have recently found different applications in cryptography. Our results show that moduli $N = p^r q$ are more susceptible to partial key exposure attacks than the original RSA moduli $N = pq$.
(Parts of the results in this chapter were published at Crypto 2003 in a joint work with Johannes Blömer [5] *and at PKC 2004* [49]*.)*

Chapter 7: Many of our new attacks in the Chapters 4–6 rely on Coppersmith's method for modular multivariate polynomial equations. In order to apply this method, one has to construct a lattice basis. However, this construction is a non-trivial task: It is not clear, which choice of the lattice basis is optimal and how to analyze certain properties of the lattice basis.

In Chapter 7, we present a method for optimizing lattice bases. Our method applies to modular bivariate polynomials of a special form but may be extended to other polynomials. The new approach can be seen as an alternative method to the so-called geometrically progressive matrices that were introduced by Boneh and Durfee [12] in 1999.
(Parts of the results in this chapter were published at the "Cryptography and Lattices Conference 2001" in a joint work with Johannes Blömer [7]*.)*

To conclude, we derive several polynomial time attacks on RSA using Coppersmith's method for finding small roots of modular polynomial equations. Interestingly, none of these attacks seems to be applicable for cryptosystems that are based on the hardness of the discrete logarithm problem. In fact, to our knowledge there are also no other attacks of this type mentioned in the cryptographic literature. At the moment, it seems that discrete logarithm based schemes are less vulnerable to these attacks. This resistance is a good reason to select discrete logarithm based schemes more often for cryptographic applications in the future.

2 RSA and Lattices

"A New Kind of Cipher that would Take Millions of Years to break"
Martin Gardner, Scientific American, 1977

"The Magic Words Are Squeamish Ossifrage"
Derek Atkins, Michael Graff, Arjen K. Lenstra, Paul C. Leyland, 1994

2.1 The RSA cryptosystem

The RSA cryptosystem has been published by Rivest, Shamir and Adleman [56] in 1978. It is the first *public key cryptosystem* (PKCS) in the literature and since its invention it has attracted a lot of attention both from the practice and from the theory of cryptography.

Let us first define the notion of a public key cryptosystem.

Definition 1 (PKCS) *A public key cryptosystem is a five-tuple* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ *satisfying the following conditions:*

1. $\mathcal{P}$ *is a finite set of possible plaintexts.*
2. $\mathcal{C}$ *is a finite set of possible ciphertexts.*
3. *The keyspace* $\mathcal{K}$ *is a finite set of possible keys.*
4. *For each* $K \in \mathcal{K}$*, there is an encryption rule* $e_K \in \mathcal{E}$, $e_K : \mathcal{P} \rightarrow \mathcal{C}$ *and a corresponding decryption rule* $d_K \in \mathcal{D}$, $d_K : \mathcal{C} \rightarrow \mathcal{P}$ *such that*

   $$d_K(e_K(m)) = m$$

   for every plaintext $m \in \mathcal{P}$.
5. *The encryption function* e_K *is public, whereas the decryption function* d_K *is secret.*

Notice that the fourth condition tells us that the decryption function d_K is the inverse of the encryption function e_K, i.e., an application of d_K yields for every ciphertext $e_K(m)$ the underlying plaintext message m.

Furthermore, we are only interested in efficient public key cryptosystems, i.e., the key $K \in \mathcal{K}$ as well as the functions e_K and d_K should be efficiently computable. One of these efficient cryptosystems is the RSA cryptosystem which we describe in the following.

Let us therefore introduce some useful notation. Let N be a positive integer. We denote by $\mathbb{Z}_N := \{0, 1, \ldots, N-1\}$ the ring of integers modulo N. The set $\mathbb{Z}_N^* \subset \mathbb{Z}_N$ consists of all integers in $\mathbb{Z}_N$ that are coprime to N, i.e., $\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid \gcd(x, N) = 1\}$. It is not hard to see that the set $\mathbb{Z}_N^*$ forms an abelian group under multiplication. Here, we only want to argue that every $x \in \mathbb{Z}_N^*$ has a multiplicative inverse. It is well-known that the Extended Euclidean Algorithm (see for instance [32], Section 3.2) computes integers a, b such that

$$ax + bN = \gcd(x, N) = 1.$$

Reducing this equation modulo N gives us the congruence $ax = 1 \bmod N$ and therefore a is the multiplicative inverse of x modulo N.

Furthermore, we denote the number of elements in $\mathbb{Z}_N^*$ by $\phi(N)$. The function $\phi(N)$ will also be called the Euler totient function. Notice that $\phi(N)$ is a multiple of the group order of $\mathbb{Z}_N^*$.

Now, we can describe the RSA scheme.

The RSA crypto-system

Let $N = pq$, where p and q are primes. Let $\mathcal{P} = \mathcal{C} = \mathbb{Z}_N$. We define the keyspace $\mathcal{K}$ as

$$\mathcal{K} := \{(N, e, d) \mid ed = 1 \bmod \phi(N)\}.$$

The public encryption function $e_K : \mathbb{Z}_N \to \mathbb{Z}_N$ is defined by

$$e_K(m) := m^e \bmod N.$$

The secret decryption function $d_K : \mathbb{Z}_N \to \mathbb{Z}_N$ is defined similarly:

$$d_K(c) := c^d \bmod N.$$

Since e_K is determined by N and e, we call (N, e) the RSA public key tuple. N is called an RSA modulus and e is called the public exponent. The secret value d is called the secret key or the secret exponent.

In order to show that the RSA system is really a PKCS, we must show that the decrypting function inverts the encryption function. Let us rewrite the equation that defines the keyspace $\mathcal{K}$ — which is also called the RSA key equation — as

$$ed = 1 + k\phi(N) \quad \text{for some } k \in \mathbb{N}.$$

Let $m \in \mathbb{Z}_N^*$. Since $\phi(N) = |\mathbb{Z}_N^*|$, we know by Euler's Theorem (see for instance [40], Proposition I.3.5) that $m^{\phi(N)} = 1 \bmod N$. Therefore, we have

$$d_K(e_K(m)) = m^{ed} = m^{1+k\phi(N)} = m \cdot \left(m^{\phi(N)}\right)^k = m \bmod N.$$

Using the Chinese Remainder Theorem one can show that the above equation holds also for every $m \in \mathbb{Z}_N \setminus \mathbb{Z}_N^*$. Thus, the RSA cryptosystem satisfies all conditions for a PKCS that are given in Definition 1.

Remark (Signatures with RSA): The RSA cryptosystem has the nice property that the functions e_K and d_K are commutative. This means that for all $m \in \mathbb{Z}_N$:

$$e_K(d_K(m)) = m.$$

Therefore, one can use RSA also as a signature scheme. It works in the following way: A user signs a message m by computing the signature $d_K(m)$ with his secret key. Now, everyone can check the validity of the signature by inverting $d_K(m)$ with the help of the public encryption function e_K and a comparison with m.

Efficiency of RSA

In order for a PKCS to be useful, we need that $K \in \mathcal{K}$ and that the functions e_K and d_K are efficiently computable. For the computation of K, we need that RSA moduli $N = pq$ can be efficiently computed, i.e., that one can efficiently select prime factors p and q. For this topic, we refer to the common textbooks on cryptography (see [50, 65]). Instead, we want to show that e_K and d_K can be computed efficiently.

Hence, we have to show that raising a message m to the e^{th} power modulo N can be done in time polynomial in $\log N$. The following algorithm is known as the Repeated Squaring Method and computes the term $m^e \bmod N$ in time $\mathcal{O}(\log e \log^2 N)$. Here we write an n-bit exponent $e = \sum_{i=0}^{n-1} e_i 2^i$ in form of its binary representation $e_{n-1} \dots e_0$.

Algorithm Repeated Squaring Method

INPUT: $m, N, e = e_{n-1} \dots e_0$

1. Set $z := 1$.
2. For $i = 0$ to $n-1$
   a) If $(e_i = 1)$ set $z := z \cdot m \bmod N$.
   b) If $(i < n-1)$ set $m := m^2 \bmod N$.

OUTPUT: $z = m^e \bmod N$

Each multiplication and squaring modulo N takes time $\mathcal{O}(\log^2 N)$. The number of squarings is $n - 1 = \lfloor \log e \rfloor$ and the number of multiplications is identical to the number of ones in the binary representation of e.

Therefore for efficiency reasons, one often uses RSA with public exponent $e = 3$ and $e = 2^{16} + 1$: The choice $e = 3$ requires 1 squaring and 2 multiplications, whereas the choice $e = 2^{16} + 1$ requires 16 squarings and 2 multiplications.

Analogous to e_K, the decryption function d_K can be computed in time $\mathcal{O}(\log d \log^2 N)$. However, in situations were decryptions/signatures are computed more often than encryptions, it would be preferable to use small decryption exponents d in order to speed up the decryption/signature computation. In another reasonable scenario, the encryptions might be done by a fast computing device (e.g. a server), whereas the decryptions/signatures are computed by a slow device (for instance a smart-card). Then, it would be preferable to shift some load to the fast device by choosing a small d with corresponding large e.

However in 1990, Wiener [71] showed that a secret key $d < \frac{1}{3}N^{\frac{1}{4}}$ yields the factorization of N in polynomial time (see also Chapter 4). Therefore, one cannot use small values of d directly.

In 1982, Quisquater and Couvreur [55] suggested an alternative fast decryption algorithm for RSA. Their trick is to compute m^d modulo p and q separately and then to combine both results using the Chinese Remainder Theorem to the value $m^d \bmod N$. Since the group order of $\mathbb{Z}_p^*$ is $p - 1$, we know that $m^d = m^{d \bmod p-1} \bmod p$. Therefore, it suffices to use the value $d_p := d \bmod p - 1$ in the exponent (and symmetrically $d_q := d \bmod q - 1$ for the computation modulo q).

Quisquater-Couvreur Method

INPUT: $m, N, p, q, d_p := d \bmod p - 1, d_q := d \bmod q - 1$

1. Compute $m^{d_p} \bmod p$ with the Repeated Squaring Method.
2. Compute $m^{d_q} \bmod q$ with the Repeated Squaring Method.
3. Compute $m^d \bmod N$ by Chinese Remaindering.

OUTPUT: $m^d \bmod N$

Let us compare the Quisquater-Couvreur approach with the Repeated Squaring Method that computes $m^d \bmod N$ directly. Here we assume that p and q are of the same bitsize, which is the normal case in RSA. For the computation $m^{d_p} \bmod p$, the modulus p has roughly half of the bits of N. Since multiplications and squarings can be done in quadratic time, they can be performed modulo p four times faster than modulo N.

Additionally, d_p has roughly half of the bits of d. Hence, the computation of $m^{d_p} \bmod p$ is about eight times faster than the computation of $m^d \bmod N$. Similarly, we can argue for the computation of $m^{d_q} \bmod q$. The running time for the Chinese Remainder step can be neglected. Hence, the Quisquater-Couvreur method is about four times faster than normal repeated squaring.

Furthermore for the Quisquater-Couvreur method, d can be chosen in $\mathbb{Z}^*_{\phi(N)}$ such that both values d_p and d_q are small. This speeds up the method even further. Therefore, small values of d_p and d_q are often used in practice. There is no polynomial time attack known if d_p and d_q are both small and the prime factors p and q have the same bit-size. However, we will present an attack for unbalanced prime factors in Chapter 5.

> *"We can factor the number 15 with quantum computers. We can also factor the number 15 with a dog trained to bark three times".*
> Robert Harley, Sci.crypt.
>
> *"The obvious mathematical breakthrough would be the development of an easy way to factor large prime numbers."*
> Bill Gates, The Road Ahead, p.265

Security of RSA:
The security of RSA is based on the assumption that one cannot invert the encryption function $e_K(m) = m^e \bmod N$ in time polynomial in $\log N$. That means, the security of RSA relies on the difficulty of taking e^{th} roots modulo some number N of unknown factorization. The hope is that e_K is a so-called *one-way permutation*. Let us define one-way permutations via one-way functions. Informally speaking, one-way functions are functions that are easy to compute but computationally infeasible to invert. Then a one-way permutation is a bijective one-way function from a set X to itself. There are many candidates for one-way functions proposed in the literature, but no function has been proven to be one-way.

An obvious way to invert $e_K(m)$ is to compute the Euler totient function $\phi(N)$: From $\phi(N)$, one can compute the secret d, which in turn determines d_K. Notice that

$$\begin{aligned} \phi(N) = |\mathbb{Z}^*_N| &= |\mathbb{Z}_N \setminus \{0, p, 2p, \ldots, (q-1)p, q, 2q, \ldots, (p-1)q\}| \\ &= N - q - (p-1) = (p-1)(q-1). \end{aligned}$$

Thus, if we knew the factors p and q, we could decrypt every message. In other words: The knowledge of the factorization of N yields $\phi(N)$ in polynomial time. Therefore, e_K is also called a candidate *trapdoor one-way permutation*. We assume that inverting e_K is computationally infeasible but with the knowledge of the trapdoor, i.e., the knowledge of the factorization of N, the inversion is easy.

The converse reduction is also true: The knowledge of $\phi(N)$ yields the factorization of N. Suppose we know $\phi(N)$ and N, then we have the following system of two equations

$$\left| \begin{array}{rcl} (p-1)(q-1) & = & \phi(N) \\ pq & = & N \end{array} \right|$$

We can easily solve this system of equations in the unknown parameters p and q. Thus the problem of determining $\phi(N)$ and the problem of factoring N are polynomial time equivalent, i.e., a polynomial time algorithm for the first problem implies a polynomial time algorithm for the second problem and vice versa.

Up to now, there is no known algorithm that computes the factorization of N in time polynomial in the bit-size of N. The best algorithm which is due to Lenstra, Lenstra and Manasse and Pollard [45] is called the Number Field Sieve and has an asymptotical running time of

$$\mathcal{O}\left(e^{(1.92+o(1))(\ln N)^{\frac{1}{3}}(\ln\ln N)^{\frac{2}{3}}}\right).$$

It is worth noticing that for quantum computers, Shor [61] presented a polynomial time algorithm for factoring N. Unfortunately (or fortunately for cryptographers!), one does not know how to build quantum computers with sufficiently many quantum bits in practice.

It is often stated in the literature that RSA is based on the factorization problem. This means that if factoring can be done in polynomial time then inverting the RSA encryption function e_K can also be done in polynomial time. However, the converse is not known: One does not know whether computing e^{th} roots modulo N is polynomial time equivalent to the factorization problem.

In 1996, Boneh and Venkatesan [17] posed some doubt whether there is a polynomial time equivalence between these problems. Namely, they showed that under so-called algebraic reductions every algorithm that factors N by using an oracle for e^{th} roots can be turned into an algorithm that does not need such an e^{th}-root oracle. Thus, the oracle does not seem to help in factoring N. The conclusion is that factoring may be harder than inverting the RSA encryption function. But at the moment one does not know how to invert RSA encryptions without knowing the factorization of N.

However, despite intensive research in the last 25 years from a mathematical point of view one does not know a more efficient way to break the RSA scheme than factoring the modulus N. Although many other attacks on RSA – which include different forms of side-channel attacks – were proposed, these approaches do not attack RSA itself but physical implementations of the RSA scheme. For instance in fault attacks, one tries to induce faults during the signature computation on the computing unit (computer, smart card, etc.). The idea behind all side channel attacks is: Since the secret key is used in the decryption/signature process, one can learn some secret information by studying the decryption/signature generation. We will give an example of a side-channel attack where an attacker learns bits of d in Section 6.1.

Let us define the following transitive relation: $A \Rightarrow B$ means that if there exists a (probabilistic) polynomial time algorithm for A, then there also exists a polynomial time algorithm for B. Up to now we have the situation

$$\text{Taking } e^{th} \text{ roots modulo } N \Leftarrow \text{Factoring } N \Leftrightarrow \text{Computing } \phi(N) \Rightarrow \text{Computing } d$$

The following theorem proves that also

$$\text{Computing } d \Rightarrow \text{Factoring } N.$$

That means we reduce the problem of factoring N in probabilistic polynomial time to the problem of determining the secret key d. However, this does not imply that taking e^{th} roots modulo N is equivalent to factoring, since there might be a way to take e^{th} roots without knowing d.

We state the following theorem for moduli $N = p^r q$ for some constant $r \geq 1$, since we will use the theorem in this form in Section 6.6. The special case $r = 1$ is the normal RSA case.

Theorem 2 *Let $N = p^r q$ with p, q prime and $r \geq 1$. Suppose we are given positive integers $e, d > 1$ satisfying*

$$ed = 1 \bmod \phi(N).$$

Then the factorization of N can be found in probabilistic polynomial time.

Proof: A complete proof of Theorem 2 for the special case $r = 1$ can be found in the book of Stinson [65]. A generalization to arbitrary r is straight-forward. Here, we only want to outline the idea of the proof.

Let $ed = 2^k t$ for some odd $t \in \mathbb{N}$. Furthermore, let us randomly choose some $m \in \mathbb{Z}_N$. We know that $m^{2^k t} = 1 \bmod N$. Therefore, $m^{2^{k-1} t}$ is a square-root of 1 modulo N. But there are four different square roots of 1 modulo N that correspond to the solutions of the congruences

$$\left| \begin{array}{rcl} x_p^2 & = & 1 \bmod p^r \\ x_q^2 & = & 1 \bmod q \end{array} \right|.$$

The four solutions are $(x_p, x_q) = (\pm 1, \pm 1)$. The solutions $(1,1)$,$(-1,-1)$ correspond to the two square roots 1,-1 of 1 modulo N, respectively. However, the square roots corresponding to the solutions $(1,-1)$ and $(-1,1)$ give us the factorization of N: Consider for instance an integer b corresponding to $(1,-1)$. Then

$$\begin{array}{rcrl} b-1 & = & 0 & \bmod p^r \quad \text{and} \\ b-1 & = & -2 & \bmod q. \end{array}$$

Therefore, we have $\gcd(N, b-1) = p^r$ which gives us p. Alternatively, we could compute $\gcd(N, b+1) = q$.

Hence, if $m^{2^{k-1}t} \neq \pm 1 \bmod N$, then a greatest common divisor computation yields the factorization of N. If $m^{2^{k-1}t} = 1 \bmod N$ then we can look at the next square root $m^{2^{k-2}t} = 1 \bmod N$, and so on. This approach will only fail if one of the square roots is -1 or if the last square root $m^t \bmod N$ is still 1. A careful analysis shows that this case happens with probability at most $\frac{1}{2}$ where the probability is taken over the random choices of m. $\square$

It is worth noticing that the hardness of inverting the RSA encryption function e_K is not sufficient at all to make RSA a cryptographically secure system. For instance, this does not exclude that it might be easy for an attacker to learn from $e_K(M)$ certain parts of the underlying message m. A convenient security requirement for public key cryptosystems is the notion of semantic security. Informally this means that an adversary is unable to distinguish between an encryption of a given plaintext and a random string.

The RSA cryptosystem as described in this chapter is not semantically secure, since the scheme is deterministic, i.e., each plaintext m has exactly one ciphertext $e_K(m)$. Furthermore, every encryption leaks the Jacobi symbol of the underlying message. However, Bellare and Rogaway [2] proposed a method called *Optimal Asymmetric Encryption Padding (OAEP)* that can be used in order to turn RSA into a semantically secure PKCS in the so-called Random Oracle Model, provided that e_K is a one-way permutation ([31], see also [10, 62]).

We do not consider semantically secure versions of RSA in this thesis, since in the subsequent chapters we will study only attacks that completely break the RSA system for keys $K \in \mathcal{K}$ of a special form. In these cases, e_K is not a trapdoor one-way permutation since the factorization of N can be found in polynomial time.

Parameter choices in RSA and useful estimates

Here, we will make a few conventions about the parameters of RSA. If not explicitly stated otherwise, we will always assume that N is a product of two different prime numbers p and q, which are of the same bit-size. This is a reasonable assumption, since for the fastest known special purpose factorization algorithm, the Elliptic Curve Method due to Lenstra [43], the running time on input N depends on the minimum of p and q. Therefore, choosing p and q of the same bit-size is a worst-case scenario for the Elliptic Curve Method.

Furthermore, we will assume without loss of generality that $p > q$. With our assumptions so far, we can conclude that

$$q < \sqrt{N} < p < 2q < 2\sqrt{N}.$$

This gives us an easy estimate for the term $p + q$ which will be frequently used in the subsequent chapters

$$p + q \leq 3\sqrt{N}. \tag{2.1}$$

Sometimes, we will need a tighter approximation of $p+q$. Let $p = c\sqrt{N}$ and $q = \frac{1}{c}\sqrt{N}$ for some $c > 1$. It is not hard to see that $p+q$ is maximal for c as large as possible. But $\frac{p}{q} < 2$ implies that $c < \sqrt{2}$. Therefore, we get the more refined bound

$$p + q \leq \frac{3}{\sqrt{2}}\sqrt{N}. \tag{2.2}$$

In order to lower bound the value $p+q$, we observe that $p+q > p > \sqrt{N}$. We obtain a better bound if we use the same reasoning as before: Let $p = c\sqrt{N}$ and $q = \frac{1}{c}\sqrt{N}$ for some $c > 1$. The sum $p+q$ is minimal for c as small as possible. Thus, we obtain

$$p + q \geq 2\sqrt{N}.$$

Since p and q are of the same bit-size and $q < \sqrt{N} < p$, the value $\sqrt{N}$ must be of the same bit-size as p and q. Thus

$$p - q \leq \sqrt{N}.$$

If not stated otherwise, we will always assume that $e \in \mathbb{Z}^*_{\phi(N)}$ which implies $e < \phi(N)$. Furthermore, we will use the inequality

$$\frac{1}{2}N < \phi(N) < N.$$

2.2 Preliminaries on lattices

"God made the integers, all else is the work of man."
Leopold Kronecker (1823-1891)

In the following, we state a few basic facts about lattices and lattice basis reduction and refer to the textbooks [22, 33, 46, 58] for a thorough introduction into the theory of lattices. In this thesis, we only use lattices that are defined over the integers but all of our definitions in this section are also valid for the real numbers.

Let $v_1, \ldots, v_n \in \mathbb{Z}^m$, $m \geq n$ be linearly independent vectors. A lattice L spanned by $\{v_1, \ldots, v_n\}$ is the set of all integer linear combinations of $v_1, \ldots, v_n$

$$L = \left\{ v \in \mathbb{Z}^m \;\middle|\; v = \sum_{i=1}^{n} a_i v_i \text{ with } a_i \in \mathbb{Z} \right\}.$$

If $m = n$, the lattice is called a full rank lattice. The set of vectors $B = \{v_1, \ldots, v_n\}$ is called a basis for L. We also say that L is spanned by the vectors of the basis B. We call $\dim(L) := n$ the dimension of L. For an example of a two-dimensional lattice with basis $B = \{(0, 2), (1, 1)\}$, see Figure 2.1.

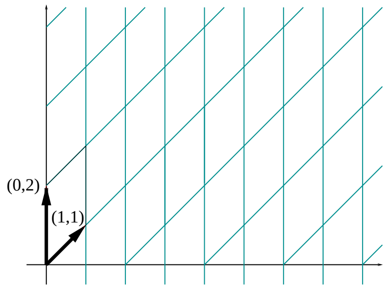

Figure 2.1: Lattice with basis $B = \{(0,2),(1,1)\}$

By $v_1^*, \ldots, v_n^*$ we denote the vectors obtained by applying Gram-Schmidt orthogonalization to the basis vectors. The determinant of L is defined as

$$\det(L) := \prod_{i=1}^{n} \|v_i^*\|,$$

where $\|v\|$ denotes the Euclidean norm of v. Any lattice L has infinitely many bases but all bases have the same determinant. If a lattice is full rank, $\det(L)$ is the absolute value of the determinant of the $(n \times n)$-matrix whose rows are the basis vectors $v_1, \ldots, v_n$.

Hence if a basis matrix is triangular, the determinant can easily be computed by multiplying the entries on the diagonal of the basis matrix. In the subsequent sections, we will mostly deal with triangular basis matrices. In Chapter 7, we will introduce non-triangular bases but we will reduce the determinant computations for these bases also to the case of triangular bases.

A well-known result by Minkowski [51] relates the determinant of a lattice L to the length of a shortest vector in L.

Theorem 3 (Minkowski) *Every n-dimensional lattice L contains a non-zero vector v with $\|v\| \leq \sqrt{n}\det(L)^{\frac{1}{n}}$.*

Unfortunately, the proof of this theorem is non-constructive.

In dimension 2, the Gauss reduction algorithm finds a shortest vector of a lattice (see for instance [58]). In arbitrary dimension, we can use the famous L^3-reduction algorithm of Lenstra, Lenstra and Lovász [44] to approximate a shortest vector. We prove an upper bound for each vector in an L^3-reduced lattice basis in the following theorem. The bounds for the three shortest vectors in an L^3-reduced basis will be frequently used in the following chapters.

Theorem 4 (Lenstra, Lenstra, Lovász) *Let $L \in \mathbb{Z}^n$ be a lattice spanned by $B = \{b_1, \ldots, b_n\}$. The L^3-algorithm outputs a reduced lattice basis $\{v_1, \ldots, v_n\}$ with*

$$\|v_i\| \leq 2^{\frac{n(n-1)}{4(n-i+1)}} \det(L)^{\frac{1}{n-i+1}} \quad \textit{for } i = 1, \ldots, n$$

in time polynomial in n and in the bit-size of the entries of the basis matrix B.

Proof: Lenstra, Lenstra and Lovász [44] showed that the L^3-algorithm outputs a basis $\{v_1, \ldots, v_n\}$ satisfying

$$\|v_i\| \leq 2^{\frac{j-1}{2}} \|v_j^*\| \quad \text{for } 1 \leq i \leq j \leq n.$$

For a vector v_i, we apply this inequality once for each j, $i \leq j \leq n$, which yields

$$\|v_i\|^{n-i+1} \leq \prod_{j=i}^{n} 2^{\frac{j-1}{2}} \|v_j^*\|. \tag{2.3}$$

Since $L \in \mathbb{Z}^n$, we know that $\|v_1^*\| = \|v_1\| \geq 1$. For an L^3-reduced basis, we have

$$2\|v_j^*\|^2 \geq \|v_{j-1}^*\|^2 \quad \text{for } 1 < j \leq n \ \ \text{(see [44])}.$$

Therefore, we obtain

$$2^{\frac{j-1}{2}} \|v_j^*\| \geq \|v_1^*\| \geq 1 \quad \text{for } 1 \leq j \leq n.$$

We apply the last inequality in order to bound

$$\prod_{j=i}^{n} 2^{\frac{j-1}{2}} \|v_j^*\| \leq \prod_{j=1}^{n} 2^{\frac{j-1}{2}} \|v_j^*\| = 2^{\frac{n(n-1)}{4}} \det(L).$$

Using this inequality together with inequality (2.3) proves the theorem. □

3 Coppersmith's method

Every integer that is sufficiently small in absolute value, must be zero.

3.1 Introduction

In 1996, Coppersmith [18, 19, 20, 21] introduced a method for finding small roots of modular polynomial equations using the L^3-algorithm. Since then, the method has found many different applications in the area of public key cryptography. Interestingly, besides many results in cryptanalysis [3, 13, 14, 16, 20, 28] it has also been used in the design of provably secure cryptosystems [62].

We will present Coppersmith's method in this chapter. Nearly all the results in the subsequent chapters will be based on Coppersmith's method. In fact, one can view our new results as different applications of the method. Since all of our applications belong to the field of cryptanalytic attacks on the RSA cryptosystem, we will first give an informal example that explains the significance of a modular root finding method in the area of public key cryptanalysis and here especially for RSA.

In public key cryptosystems, we have a public key/secret key pair. For instance in the RSA scheme, the public key is the tuple (N, e) and the secret key is d. The public key/secret key pair satisfies

$$ed = 1 + k\phi(N), \quad \text{where } k \in \mathbb{N}.$$

We see that in the RSA scheme, the public key pair (N, e) satisfies a relation with the unknown parameters d, k and $\phi(N) = N - (p + q - 1)$. Hence, we can assign the unknown values d, k and $p+q-1$ the variables x, y and z, respectively. Then we obtain a polynomial

$$f(x, y, z) = ex - y(N - z) - 1$$

with the root $(x_0, y_0, z_0) = (d, k, p + q - 1)$ over the integers. Suppose we could find the root (x_0, y_0, z_0). Then we could solve the following system of two equations in the unknowns p and q

$$\left| \begin{array}{rcl} p + q - 1 & = & z_0 \\ pq & = & N \end{array} \right|.$$

Substituting $q = \frac{N}{p}$ in the first equation gives us the quadratic equation $p^2 - (z_0+1)p + N = 0$ which has the two solutions p and q.

Hence, finding the root (x_0, y_0, z_0) is polynomial time equivalent to the factorization of N. Unfortunately, we do not know how to find such a root and since we assume that factoring is hard, we cannot hope to find an efficient method for extracting (x_0, y_0, z_0) in general. However, the inability of finding a general solution does not imply that there are no roots (x_0, y_0, z_0) of a *special form* for which we can solve the problem in polynomial time in the bit-length of N. This *special form* is the point where Coppersmith's method comes into the play.

Suppose we had a general method how to find *small* roots of modular polynomial equations, where *small* means that every component of the root is small compared to the modulus. Could we then discover the unknown solution (x_0, y_0, z_0) of the equation $f(x, y, z) = 0$? The obvious problem with this approach is:

> The equation $f(x, y, z) = 0$ is not a modular polynomial equation but an equation over the integers.

It is easy to come around with this problem: Simply choose one of the known parameters in $f(x, y, z)$ as the modulus, e.g., we could choose either N or e. Then we obtain the following modular polynomial equations, respectively:

$$f_N(x, yz) = ex + yz - 1 \quad \text{and} \quad f_e(y, z) = y(N - z) + 1.$$

We see that in f_N we can treat yz as a new variable. So in fact, both polynomials are bivariate polynomials. The polynomial f_N has the root $(d, k(p+q-1))$ modulo N, whereas the polynomial f_e has the root $(k, p+q-1)$ modulo e. Now the requirement that the roots of f_N and f_e should be small compared to the modulus makes sense.

Notice that we have introduced a notation, which we will keep throughout this thesis: The subscript N of the polynomial f_N denotes that f_N is evaluated modulo N.

In order to bound the parameter k we observe that

$$k = \frac{ed-1}{\phi(N)} < \frac{e}{\phi(N)} d < d.$$

Hence, if we choose d to be a small secret value then k is automatically small. Since the running time of RSA decryption with the Repeated Squaring Method of Section 2.1 is $\mathcal{O}(\log d \cdot \log^2 N)$, small values of d speed up the RSA decryption/signature process. Therefore, it is tempting to use small values of d for efficiency reasons.

But small choices of d lead to the two most famous attacks on RSA up to now: Wiener [71] showed in 1990 that values of $d < \frac{1}{3}N^{\frac{1}{4}}$ yield the factorization of N. In 1999, Boneh and Durfee [12] improved this result to $d < N^{1-\frac{1}{\sqrt{2}}} \approx N^{0.292}$. Interestingly, both methods can be seen as an application of Coppersmith's method:

- Wiener's method corresponds to an application of Coppersmith's method using the polynomial $f_N(x, yz)$. Originally, Wiener presented his results in terms of continued fractions. We show in Section 4.2 how an application of Coppersmith's method using the polynomial f_N leads to a two-dimensional lattice L. The shortest vector in L then yields the factorization of N, whenever the Wiener-bound $d < \frac{1}{3}N^{\frac{1}{4}}$ holds.

- The attack on RSA with small secret exponent d due to Boneh-Durfee makes use of the polynomial $f_e(y, z)$ in combination with Coppersmith's method. This leads to the improved bound of $d < N^{0.292}$.

In this thesis, we study further applications of Coppersmith's method. All these applications have the property that d is not small itself but satisfies a relation with small unknown parameters:

- A generalization of Wiener's attack: We present a new method that works if d has a decomposition $d = \frac{d_1}{d_2} \bmod \phi(N)$ into small d_1, d_2. As an application of this new approach, we cryptanalyze an RSA-type scheme presented in 2001 [73, 74].

- Attacks on unbalanced RSA with small CRT-exponent: Here d has a Chinese Remainder Theorem decomposition $d = (d_p, d_q)$, where $d_p = d \bmod p - 1$ is small. This CRT-property of d is of important practical significance, since one can use small values of d_p, d_q in order to speed up the fast Quisquater-Couvreur variant of RSA decryption (see Section 2.1). We show that the factorization of N can be found whenever the prime factor p is sufficiently large.

- Partial key exposure attacks on RSA: We present attacks on RSA when parts of the secret key d are known. For instance suppose that an attacker knows a value $\tilde{d}$ that corresponds to the k least significant bits of d, e.g. $d = x \cdot 2^k + \tilde{d}$ for some unknown x. The more bits we know, the smaller is the value of the unknown x. Thus, one can hope to recover x using Coppersmith's method whenever sufficiently many bits of d are known.

For the rest of this chapter, we present Coppersmith's method. Therefore, we review in Section 6 Coppersmith's method in the case of univariate modular polynomial equations. In the univariate case, Coppersmith's method provably outputs every root that is sufficiently small in absolute value. As an important application of the univariate case, we show a result that was also presented in Coppersmith's work in 1996 [18]: The knowledge of half of the most significant bits of p suffice to find the factorization of an RSA-modulus $N = pq$ in polynomial time. Originally, Coppersmith used a bivariate polynomial equation to show this result. However as we will see, the univariate modular approach already suffices.

In Section 3.4, we present Coppersmith's extensions to the multivariate polynomial case. The main drawback of the generalization to the multivariate case is that the method is no longer rigorous. It depends on some (reasonable) heuristic.

3.2 The univariate case

> *"Everything should be made as simple as possible, but not simpler."*
> Albert Einstein (1879-1955)

Let us first introduce some helpful notations. Let $f(x) := \sum_i a_i x^i$ be a univariate polynomial with coefficients $a_i \in \mathbb{Z}$. We will often use the short-hand notation f for $f(x)$. All terms x^i with non-zero coefficients are called monomials. The degree of f is defined as $\max\{i \mid a_i \neq 0\}$. Let f be a polynomial of degree i, then we call a_i the leading coefficient of f. A polynomial is called monic if its leading coefficient is 1. The coefficient vector of f is defined by the vector of the coefficients a_i. We define the norm of f as the Euclidean norm of the coefficient vector: $\|f\|^2 := \sum_i a_i^2$. The definitions for multivariate polynomials are analogous.

In this section, we study the following problem:

> Let N be a composite number of unknown factorization and b be a divisor of N, where $b \geq N^\beta$ for some known β. Notice that b must not be a prime. Given an univariate polynomial $f_b(x) \in \mathbb{Z}[X]$ of degree δ. Find in time polynomial in δ and in the bit-size of N all roots $x_0 \in \mathbb{Z}$ of $f_b(x)$ modulo b, where $|x_0| < X$ for some X. The goal is to choose the upper bound X for the size of the roots x_0 as large as possible.

First, we consider one important special case of the problem above: The case, where $b = N$. We want to point out that this is the case that was studied by Coppersmith in his original work [20]. However, we think it is convenient to formulate Coppersmith's method in a more general framework, since in this more general form we can easily derive important applications of Coppersmith's method in the following sections as simple implications of the results of the problem above. For instance, some results in the literature — e.g. the factorization algorithm for moduli $N = p^r q$ with large r due to Boneh, Durfee and Howgrave-Graham [16] — use Coppersmith's method but do not fit in the original formulation of the problem. To our knowledge, the new formulation covers all approaches mentioned in the literature.

Nevertheless, let us first give some motivation for the importance of the case $b = N$. We do not know how to solve modular polynomial equations, when the factorization of the modulus N is unknown. Notice that the problem of decrypting an RSA-encrypted message $c = m^e \bmod N$ is the problem of finding the unique positive root $x_0 = m < N$

of the polynomial

$$f_N(x) = x^e - c \bmod N.$$

Under the assumption that inverting the RSA-function is hard, we cannot solve this problem in general. If we knew the factors p and q, we could find the roots of $f_N(x)$ modulo p and modulo q. Using the Chinese Remainder Theorem, we could then combine both solutions and recover x modulo N.

But if we cannot solve the problem for all $m \in \mathbb{Z}_N$, then it might be feasible for especially small values of m. Indeed, it is a well-known protocol failure of RSA that one can recover m in polynomial time whenever $m < N^{\frac{1}{e}}$. The reason why this attack works is simple: Since $m^e < N$, we have

$$m^e - c = 0 \ \text{ over } \mathbb{Z}$$

and not just modulo N. Thus, we can simply take the e^{th} root of c in order to recover the value of m. Notice that this attack can easily be avoided by padding each message m with additional bits such that m becomes sufficiently large.

However, the attack above already yields a simple idea how to solve modular univariate polynomial equations:

> Reduce the root finding problem in modular equations to the case of root finding in equations over the integers.

Thus, the idea is to construct from the polynomial $f_b(x)$ with the root $x_0 \leq X$ modulo b a polynomial $f(x)$ which has the same root x_0 over the integers. Then we can easily find the root x_0 by applying standard root finding algorithms to $f(x)$ (take for instance the Sturm sequence, see [39]).

But how can we transform $f_b(x)$ into $f(x)$? This transformation is exactly the core of Coppersmith's method. In the following, we describe Coppersmith's reduction of the modular root finding case to root finding case over the integers.

Reduction of the modular to the integer case
Recall that we start with the knowledge of the following values:

Known parameters:
- the modulus N of unknown factorization with divisor b
- the lower bound N^β with $b \geq N^\beta$
- the polynomial $f_b(x)$ with $f_b(x_0) = 0 \bmod b$ for some $|x_0| \leq X$

Unknown parameters:
- the root x_0
- the modulus b.

In order to construct $f(x)$, we fix an integer m and construct a set G of univariate polynomials where each polynomial $g(x) \in G$ satisfies

$$g(x_0) = 0 \bmod b^m.$$

Since we do not know the values of x_0 and b, we can only use the parameters $f_b(x)$ and N in order to construct the set G. This means that we can choose polynomials of the form

$$g_{i,j}(x) = N^{m-i} x^j f_b^i(x) \quad \text{for } i = 1, \ldots, m \text{ and some choice of } j.$$

Notice that for each choice of i, j we have $g_{i,j}(x_0) = 0 \bmod b^m$ since $f_b^i(x_0) = 0 \bmod b^i$ and N^{m-i} is a multiple of b^{m-i}.

But then every integer linear combination

$$f(x) = \sum_{i,j} a_{i,j} g_{i,j}(x) \ , \quad a_{i.j} \in \mathbb{Z} \tag{3.1}$$

of polynomials in G also has the root x_0 modulo b^m. Our goal is to find among these linear combinations one which has the root x_0 not just modulo b^m but also over the integers. But which property does the above linear combination $f(x)$ of polynomials in G need to satisfy in this case? The most simple idea is to choose the coefficients $a_{i,j}$ such that $f(x)$ satisfies the relation

$$|f(x_0)| < b^m.$$

Since we know that $f(x_0) = 0 \bmod b^m$ and $f(x_0) \in \mathbb{Z}$, we can conclude that $f(x_0) = 0$ over the integers. In other words we use the following basic principle that underlies Coppersmith's method:

Every integer multiple of b^m that is small enough must be zero!

However, there remain two serious problems with the approach so far:

1. How can we find the coefficients $a_{i,j}$ efficiently?

2. How can we check that the relation $|f(x_0)| < b^m$ is satisfied if we do not know the values of x_0 and b?

Let us first address the second problem. Observe that instead of x_0 and b we know bounds X, N^β satisfying $|x_0| \leq X$ and $b \geq N^\beta$. Thus, instead of checking if a polynomial $f(x)$ is smaller than b^m at the point x_0, we could check whether the condition $|f(X)| < N^{\beta m}$ holds. In other words: We check if f evaluated at a *larger* point gives some *smaller* value. But still this approach makes some problems: Let $f(x) := \sum_{i=0}^{n-1} c_i x^i$, $c_i \in \mathbb{Z}$. We could be in the case, where we have $|f(X)| < N^{\beta m}$ and still $|f(x_0)| \geq b^m$. The reason for that is that the coefficients c_i may be negative. Some terms in $f(X)$ may cancel out where in $f(x_0)$ they lead to a large value.

Hence, we cannot evaluate $f(x)$ at the point X directly but we first have to force all coefficients c_i of $f(x)$ to be non-negative by taking the absolute values. Using the triangle inequality, we obtain the following inequalities:

$$|f(x_0)| \leq \sum_{i=0}^{n-1} |c_i x_0^i| \leq \sum_{i=0}^{n-1} |c_i| X^i.$$

If we could satisfy the inequality $\sum_{i=0}^{n-1} |c_i| X^i < N^{\beta m}$ this would automatically imply the desired condition $|f(x_0)| < N^{\beta m} \leq b^m$. We observe that the polynomial

$$f(xX) = \sum_{i=0}^{n-1} c_i (xX)^i$$

has the coefficient vector $(c_0 X^0, c_1 X^1, \ldots, c_{n-1} X^{n-1})$. Thus, the term $\sum_{i=0}^{n-1} |c_i| X^i$ is the ℓ_1-norm of the coefficient vector of $f(xX)$ and thus by our definition of the norm of a polynomial $\|f(xX)\|_{\ell_1} = \sum_{i=0}^{n-1} |c_i| X^i$. Therefore, our polynomial $f(x)$ has to satisfy the condition $\|f(xX)\|_{\ell_1} < b^m$. Since by switching to the Euclidean norm, we loose at most a factor of $\sqrt{n}$, we could also use the condition $\sqrt{n} \cdot \|f(xX)\| < b^m$.

The last condition will be used in the following theorem. We want to point out that Coppersmith originally did not formulate this condition in terms of polynomial arithmetic. This convenient and very useful formulation is due to Howgrave-Graham [35] who revisited Coppersmith's method. Therefore, the following theorem is due to Howgrave-Graham although it resembles Coppersmith's original idea. We think that the different point of view on Coppersmith's method due to Howgrave-Graham was one of the reasons that Coppersmith's method has found so many applications in cryptography. Therefore throughout the thesis, we always refer to Howgrave-Graham's theorem by keeping in mind that it is a reformulation of Coppersmith's underlying idea.

Theorem 5 (Howgrave-Graham) *Let $f(x)$ be an univariate polynomial with n monomials. Further, let m be a positive integer. Suppose that*

(1) $f(x_0) = 0 \bmod b^m$ *where* $|x_0| < X$

(2) $\|f(xX)\| < \frac{b^m}{\sqrt{n}}$

Then $f(x_0) = 0$ *holds over the integers.*

Proof: We have

$$\begin{aligned} |f(x_0)| &= \sum_i c_i x_0^i \leq \sum_i |c_i x_0^i| \\ &\leq \sum_i |c_i| X^i \leq \sqrt{n} \|f(xX)\| < b^m. \end{aligned}$$

But $f(x_0)$ is a multiple of b^m and therefore it must be zero. $\square$

Up to this point, we only derived conditions under which a polynomial $f(x)$ has a root x_0, $|x_0| \leq X$ over the integers. Now, we want to show how to construct such a polynomial $f(x)$. Remember from equation (3.1) that every $f(x)$ which is an integer linear combination

$$f(x) = \sum_{i,j} a_{i,j} g_{i,j}(x)$$

of the polynomials $g_{i,j} \in G$ satisfies condition (1) of Howgrave-Graham's theorem, since every $g_{i,j}$ satisfies condition (1). Therefore, we have to search among all integer linear combinations for a polynomial $f(x)$ which also satisfies the second condition of Theorem 5. In other words, we have to search among all integer linear combinations of the coefficient vectors $g_{i,j}(xX)$ for one vector with Euclidean norm smaller than $\frac{b^m}{\sqrt{n}}$. But this means:

> Find in the lattice L that is spanned by the coefficient vectors of $g_{i,j}(xX)$ a vector with Euclidean norm smaller than $\frac{b^m}{\sqrt{n}}$.

Our goal is to ensure that the L^3-algorithm finds a vector v with $\|v\| < \frac{b^m}{\sqrt{n}}$ in L. Then we could apply the L^3-algorithm to the basis spanned by the coefficient vectors of $g_{i,j}(xX)$ and find a sufficiently small vector. Notice that by Theorem 4, the norm of a shortest vector v in an L^3-reduced integer basis can by related to the determinant of the corresponding lattice L with dimension n by

$$\|v\| \leq 2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}}.$$

Thus if we could satisfy the inequality

$$2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}} < \frac{N^{\beta m}}{\sqrt{n}} \tag{3.2}$$

then we have the desired inequality $\|v\| < \frac{N^{\beta m}}{\sqrt{n}} \leq \frac{b^m}{\sqrt{n}}$, which guarantees that the shortest vector of an L^3-reduced basis is sufficiently short to satisfy the second condition in

Theorem 5. Notice that inequality (3.2) gives us a very useful condition, since we can compute from the coefficient vectors of $g_{i,j}(xX)$ the determinant of L.

Furthermore, we observe that $\det(L)$ depends on the values of N and X, since the $g_{i,j}$ depend on these parameters. Thus, inequality (3.2) gives us a restriction on the size of X which can be used in Coppersmith's method. Remember that X is an upper bound for the size of the roots x_0 that can be found.

In our application of Coppersmith's method, we will often be in the case where N is very large (e.g. in the order of 2^{1000}), whereas the lattice dimension n is negligible compared to N (in the order of 100). Therefore, we can often neglect the terms that depend only on n, which simplifies condition (3.2) to the elegant form

$$\det(L) < N^{\beta mn}. \tag{3.3}$$

Since we want to maximize the size of X, we have to choose the $g_{i,j}$ such that $\det(L)$ is as small as possible. We can give an even more quantitative statement which $g_{i,j}$ are helpful in order to optimize the size of X:

Suppose we have a basis that satisfies inequality (3.3). Assume we assign a new coefficient vector of some $g_{i,j}(xX)$ to the basis. This increases the lattice dimension n by 1, which in turn increases the right-hand side of inequality (3.3) by a factor of $N^{\beta m}$. Hence, if the contribution of the new vector to the determinant is at most a factor of $N^{\beta m}$, then inequality (3.3) still holds. Moreover, if the contribution to $\det(L)$ is significantly less than $N^{\beta m}$, then we can slightly increase $\det(L)$ and the condition (3.3) will still be valid. But the ability to increase the determinant means that we can increase the value of X. Hence:

> Every basis vector that contributes to $\det(L)$ with a factor smaller than $N^{\beta m}$ is helpful !

This criterion will be used throughout the thesis in order to optimize the choices of basis vectors. Normally, we will use lattice bases whose bases matrices are in lower triangular form. In this case, the criterion means that every basis vector with diagonal entry smaller than $N^{\beta m}$ is helpful, but the criterion also holds for non-triangular bases.

Let us briefly summarize Coppersmith's method for the univariate case in some informal way. We will later specify the values of the parameters that are used here.

Coppersmith Method in the univariate case (informally)

INPUT: – Polynomial $f_b(x)$ of degree δ

– Modulus N of unknown factorization which is a multiple of b and a lower bound $b \geq N^\beta$

Step 1: Fix an integer m (depending on N, β and δ) and construct a set G of univariate polynomials, where each $g_i(x) \in G$ satisfies

$$g(x) = 0 \bmod b^m.$$

Step 2: Construct the lattice L from the polynomials $g_i(x) \in G$: The basis vectors of L are the coefficient vectors of $g_i(xX)$, where X is chosen such that the L^3-algorithm finds a vector with norm smaller than $\frac{b^m}{\dim(L)}$ in L. The size of X is a function of the parameters N, β and δ.

Step 3: Apply the L^3-algorithm to the lattice bases. Let v be the shortest vector in the L^3-reduced bases. The vector v is the coefficient vector of some polynomial $f(xX)$. Construct $f(x)$ from v.

OUTPUT: Polynomial $f(x)$ with $f(x_0) = 0$ over $\mathbb{Z}$ whenever $f_b(x_0) = 0 \bmod b$ and $|x_0| \leq X$

Notice that every step can be done in time polynomial in the input size.

We can find all roots of $f(x)$ in $\mathbb{Z}$ by a standard root finding algorithm. We want to point out that $f(x)$ may have more roots in $\mathbb{Z}$ than $f_b(x)$ modulo b. However, the number of roots is bounded by the degree of $f(x)$ and the desired roots of $f_b(x)$ with $|x_0| \leq X$ must be among the roots of $f(x)$.

Coppersmith's Theorem

Now, we will apply Coppersmith's method on arbitrary univariate, monic polynomials $f_b(x)$ of degree δ. Normally, the property that $f_b(x)$ is monic is no restriction in practice. Assume that $f_b(x)$ has a leading coefficient $a_\delta \neq 1$, then one could compute the inverse a_δ^{-1} of a_δ modulo N. Since N is a multiple of b, the value a_δ^{-1} will also be an inverse of a_i modulo b and we can make $f_b(x)$ monic by multiplying with this inverse. The only problem that might occur is the case when a_δ and the modulus N share a non-trivial greatest common divisor, since then a_δ^{-1} does not exist. However, if we assume that N is hard to factor (which will always be the case in our applications), then either this case does not occur or we have found a non-trivial factor of N.

The following theorem of Coppersmith states that for a monic polynomial $f_b(x)$ of degree δ, all roots x_0 with $|x_0| \le N^{\frac{\beta^2}{\delta}}$ can be found in polynomial time.

Theorem 6 (Coppersmith) *Let N be an integer of unknown factorization, which has a divisor $b \ge N^{\beta}$. Furthermore, let $f_b(x)$ be an univariate, monic polynomial of degree δ. Then we can find all solutions x_0 for the equation $f_b(x) = 0 \bmod b$ with*

$$|x_0| \le \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$$

in time polynomial in $(\log N, \delta, \frac{1}{\epsilon})$.

Proof: Define $X := \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$. Now, we successively apply the steps of Coppersmith's method. In the first step, we fix an integer m such that

$$m \ge \max\left\{\frac{\beta^2}{\delta\epsilon}, \frac{7\beta}{\delta}\right\}. \tag{3.4}$$

Notice that for $\epsilon \le \frac{1}{7}\beta$ this implies that we can use the value

$$m = \left\lceil \frac{\beta^2}{\delta\epsilon} \right\rceil.$$

Next, we choose a set G of polynomials, where each polynomial has a root x_0 modulo b^m whenever $f_b(x)$ has the root x_0 modulo b. In our case, we include in G the polynomials

$$\begin{array}{lllll}
N^m, & xN^m, & x^2N^m, & \dots & x^{\delta-1}N^m, \\
N^{m-1}f, & xN^{m-1}f, & x^2N^{m-1}f, & \dots & x^{\delta-1}N^{m-1}f, \\
N^{m-2}f^2, & xN^{m-2}f^2, & x^2N^{m-2}f^2, & \dots & x^{\delta-1}N^{m-2}f^2, \\
\vdots & \vdots & \vdots & & \vdots \\
Nf^{m-1}, & xNf^{m-1}, & x^2Nf^{m-1}, & \dots & x^{\delta-1}Nf^{m-1}.
\end{array}$$

Additionally, we take the polynomials

$$f^m, xf^m, x^2f^m, \dots, x^{t-1}f^m$$

for some t that has to be optimized as a function of m.

Note that by our ordering the k^{th} polynomial of G is a polynomial of degree k. Thus, it introduces the new monomial x^k. We could also write the choice of our polynomials in G in a more compact form. Namely, we have chosen the polynomials

$$\begin{array}{rclll}
g_{i,j}(x) & = & x^j N^i f^{m-i}(x) & \text{for} & i = 0, \dots, m-1,\ j = 0, \dots, \delta - 1 \text{ and} \\
h_i(x) & = & x^i f^m(x) & \text{for} & i = 0, \dots, t-1.
\end{array}$$

In Step 2 of Coppersmith's method, we construct the lattice L that is spanned by the coefficient vectors of $g_{i,j}(xX)$ and $h_i(xX)$. As we noticed before, we can order the polynomials $g_{i,j}$ and h_i in strictly increasing order of their degrees k, i.e., each polynomial introduces exactly one new monomial x^k. Therefore, the basis B of L, that has as row vectors the coefficient vectors of $g_{i,j}(xX)$ and $h_i(xX)$, has the property that it can be written as a lower triangular matrix. Let $n := \delta m + t$, then we write B as the following $(n \times n)$-matrix[1]:

$$\left(\begin{array}{ccccccccccccc}
N^m &&&&&&&&&&&& \\
& N^m X &&&&&&&&&&& \\
&& \ddots &&&&&&&&&& \\
&&& N^m X^{\delta-1} &&&&&&&&& \\
\hline
& \ddots & \ddots & \ddots & \ddots &&&&&&&& \\
\hline
- & - & \dots & - & \dots & N X^{\delta m-\delta} &&&&&&& \\
& - & \dots & - & \dots & - & N X^{\delta m-\delta+1} &&&&&& \\
&& \ddots & \ddots & \ddots & \ddots & \ddots & \ddots &&&&& \\
&&& - & \dots & - & - & \dots & N X^{\delta m-1} &&&& \\
\hline\hline
- & - & \dots & - & \dots & - & - & - & - & X^{\delta m} &&& \\
& - & \dots & - & \dots & - & - & - & - & - & X^{\delta m+1} && \\
&&&& \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
&&&&& - & - & - & - & - & - & \dots & X^{\delta m+t-1}
\end{array}\right)$$

Note that we can easily compute the determinant of L since B is in lower triangular form. Therefore, $\det(L)$ is simply the product of all entries on the diagonal:

$$\det(L) = N^{\frac{1}{2}\delta m(m+1)} X^{\frac{1}{2}n(n-1)}. \tag{3.5}$$

Now we want to optimize the parameter t, which is equivalent to the optimization of $n = \delta m + t$. Remember that in the discussion for Coppersmith's method, we argued that every vector which contributes to the determinant by a factor less than b^m is helpful. In our setting this means that we have to ensure that the entries of the coefficient vectors $h_i(xX)$ on the diagonal are all less than b^m, i.e., we have the condition

$$X^{n-1} < b^m.$$

Since $X^{n-1} < N^{(\frac{\beta^2}{\delta}-\epsilon)(n-1)} < N^{\frac{\beta^2}{\delta}n}$ and $b \geq N^\beta$ this condition is satisfied for the choice

$$n \leq \frac{\delta}{\beta} m.$$

According to (3.4), we can choose m as the maximum of $\left\{\frac{\beta^2}{\delta\epsilon}, \frac{7\beta}{\delta}\right\}$ (here we neglect roundings). Then, we immediately have a bound for the lattice dimension

$$n = \max\left\{\frac{\beta}{\epsilon}, 7\right\},$$

[1] Here we use the following notation: Every non-specified entry is zero. The entries marked with "-" may be non-zero, but the determinant of the lattice does not depend on their values

which is polynomial in $\frac{1}{\epsilon}$. Since the bit-size of the entries in B can be bounded by $(\delta + m)\log N \leq (\delta + n)\log N$, the L^3-algorithm operated on B in time polynomial in $\log N$, δ and $\frac{1}{\epsilon}$. Now, we want to prove that the L^3-algorithm also finds a vector which is sufficiently short.

In order to apply the theorem of Howgrave-Graham (Theorem 5), we have to ensure that the L^3-algorithm finds a vector in L with norm smaller than $\frac{b^m}{\sqrt{n}}$. Since the L^3-algorithm finds a vector v in an n-dimensional lattice with $\|v\| \leq 2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}}$, we have to satisfy the condition

$$2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}} < \frac{b^m}{\sqrt{n}}.$$

Using the term for $\det(L)$ in (3.5) and the fact $b \geq N^{\beta}$, we obtain the new condition

$$N^{\frac{\delta m(m+1)}{2n}} X^{\frac{n-1}{2}} \leq 2^{-\frac{n-1}{4}} n^{-\frac{1}{2}} N^{\beta m}.$$

This gives us a condition on the size of X:

$$X \leq 2^{-\frac{1}{2}} n^{-\frac{1}{n-1}} N^{\frac{2\beta m}{n-1} - \frac{\delta m(m+1)}{n(n-1)}}.$$

Notice that $n^{-\frac{1}{n-1}} = 2^{-\frac{\log n}{n-1}} \geq 2^{-\frac{1}{2}}$ for $n \geq 7$. Therefore, our condition simplifies to

$$X \leq \frac{1}{2} N^{\frac{2\beta m}{n-1} - \frac{\delta m(m+1)}{n(n-1)}}.$$

Remember that we made the choice $X = \frac{1}{2} N^{\frac{\beta^2}{\delta} - \epsilon}$. Hence in order to finish the proof of the theorem, it suffices to show that

$$\frac{2\beta m}{n-1} - \frac{\delta m^2 (1 + \frac{1}{m})}{n(n-1)} \geq \frac{\beta^2}{\delta} - \epsilon.$$

We obtain a lower bound for the left-hand side by multiplying with $\frac{n-1}{n}$. Then, we substitute $n = \frac{\delta}{\beta} m$ which gives us

$$2\frac{\beta^2}{\delta} - \frac{\beta^2}{\delta}\left(1 + \frac{1}{m}\right) \geq \frac{\beta^2}{\delta} - \epsilon.$$

This simplifies to

$$-\frac{\beta^2}{\delta} \cdot \frac{1}{m} \geq -\epsilon.$$

This in turn gives as the condition $m \geq \frac{\beta^2}{\delta \epsilon}$, which holds by the choice of m that we made in (3.4).

Let us briefly summarize the whole algorithm which finds all roots of $f_b(x)$ modulo b that are in absolute value smaller than X.

Coppersmith's method in the univariate case

INPUT: – Polynomial $f_b(x)$ of degree δ.

– Modulus N of unknown factorization which is a multiple of b and a lower bound $b \geq N^\beta$

Step 1: Choose the smallest integer m such that $m \geq \max\left\{\frac{\beta^2}{\delta\epsilon}, \frac{7\beta}{\delta}\right\}$.

Compute $t = \lfloor \delta m(\frac{1}{\beta} - 1)\rfloor$.

Compute the polynomials

$$\begin{array}{lcll} g_{i,j}(x) & = & x^j N^i f^{m-i}(x) & \text{for} \quad i = 0, \ldots, m-1,\ j = 0, \ldots, \delta - 1 \text{ and} \\ h_i(x) & = & x^i f^m(x) & \text{for} \quad i = 0 \ldots t-1. \end{array}$$

Step 2: Compute the bound $X = \lceil N^{\frac{\beta^2}{\delta} - \epsilon}\rceil$. Construct the lattice basis B, where the basis vectors of B are the coefficient vectors of $g_{i,j}(xX)$ and $h_i(xX)$.

Step 3: Apply the L^3-algorithm to the lattice bases B. Let v be the shortest vector in the L^3-reduced bases. The vector v is the coefficient vector of some polynomial $f(xX)$. Construct $f(x)$ from v.

Step 4: Find the set R of all roots of $f(x)$ over the integers. For every root $x_0 \in R$ check whether $\gcd(N, f_b(x_0)) \geq N^\beta$. If this condition is not satisfied then remove x_0 from R.

OUTPUT: Set R, where $x_0 \in R$ whenever $f_b(x_0) = 0 \bmod b$ for an $|x_0| \leq X$.

As we noticed before, all steps of the algorithm can be done in time polynomial in $\log N$, δ and $\frac{1}{\epsilon}$, which concludes the proof of the theorem. □

One should notice that the polynomial $f(x)$ that we construct in Coppersmith's method may contain integer roots that are not roots of $f_b(x)$ modulo b. It is easy to see that one could add an additional integer root $a \in \mathbb{Z}$ to $f(x)$ by multiplying with the linear term $(x - a)$. Therefore, we use in Step 4 of the algorithm in Theorem 6 a simple test whether $f_b(x_0)$ is really a divisor of N of size at least N^β. In the case where b is prime and $\beta \geq \frac{1}{2}$, b is the unique divisor of N with $b \geq N^\beta$. Therefore in this case, the set R of integer roots of $f(x)$ contains exactly the roots of $f_b(x)$ modulo b. If $\beta < \frac{1}{2}$, R may contain roots of $f_b(x)$ modulo some other divisor $q > N^\beta$ of N. Since we do not know b, we might have problems to identify the roots of $f_b(x)$ modulo b. However, for all of our applications it will suffice to know only one small root of $f_b(x)$ modulo b and this property will be easily testable.

It is also worth noticing the following point: The approximation factor of $2^{\frac{n-1}{4}}$ for the shortest vector coming from the L^3-algorithm is exponentially in the lattice dimension n, but this factor essentially translates in the analysis of Theorem 6 to the term $\frac{1}{2}$ for the upper bound of the size of the roots x_0. Thus, computing a shortest vector instead of an L^3-approximate version would only improve the bound by a factor of roughly 2 (i.e., only one bit).

Moreover, the following theorem is a direct implication of Theorem 6 and shows how we can easily remove the terms $\frac{1}{2}$ and ϵ from the upper bound on x_0 by a simple brute-force search.

Theorem 7 (Coppersmith) *Let N be an integer of unknown factorization, which has a divisor $b \geq N^{\beta}$. Let $f_b(x)$ be an univariate, monic polynomial of degree δ. Furthermore, let c_N be a function that is upper-bounded by a polynomial in $\log N$. Then we can find all solutions x_0 for the equation $f_b(x) = 0 \bmod b$ with*

$$|x_0| \leq c_N N^{\frac{\beta^2}{\delta}}$$

in time polynomial in $(\log N, \delta)$.

Proof: An application of Theorem 6 with the parameter choice $\epsilon = \frac{1}{\log N}$ shows that we can find all roots x_0 with

$$|x_0| \leq \frac{1}{4} N^{\frac{\beta^2}{\delta}}$$

in time polynomial in $\log N$ and δ.

In order to find all roots that are of size at most $c_N N^{\frac{\beta^2}{\delta}}$ in absolute value, we divide the interval $[-c_N N^{\frac{\beta^2}{\delta}}, c_N N^{\frac{\beta^2}{\delta}}]$ into $4c_N$ subintervals of size $\frac{1}{2}N^{\frac{\beta^2}{\delta}}$ centered at some x_i. For each subinterval with center x_i, we apply the algorithm of Theorem 6 to the polynomial $f_b(x - x_i)$ and output the roots in this subinterval. $\square$

For completeness reasons and since it is one of the most interesting cases of Coppersmith's method, we explicitly state the special case $b = N$ and $c_N = 1$, which is given in the work of Coppersmith [20].

Theorem 8 (Coppersmith) *Let N be an integer of unknown factorization. Furthermore, let $f_N(x)$ be an univariate, monic polynomial of degree δ. Then we can find all solutions x_0 for the equation $f_N(x) = 0 \bmod N$ with*

$$|x_0| \leq N^{\frac{1}{\delta}}$$

in time polynomial in $(\log N, \delta)$.

In the following section, we state some of the most important applications of Coppersmith's method when applied to the RSA cryptosystem.

3.3 Applications of Coppersmith's method in the univariate case

"How thoroughly it is ingrained in mathematical science that every real advance goes hand in hand with the invention of sharper tools and simpler methods which, at the same time, assist in understanding earlier theories and in casting aside some more complicated developments".
David Hilbert (1862-1943)

Attacking RSA with small e by knowing parts of the message:
In the introduction to this section, we showed an easy protocol failure of the RSA cryptosystem with small encryption exponent e. Assume that we encode a message $m < N^{\frac{1}{e}}$. Since for the ciphertext $c = m^e < N$ the modulo N reduction does not affect the computation of c, we can easily recover m by computing the e^{th} root of c over the integers.

Now consider the following problem:

> Suppose that $m = M + x$ for some known part M of the message and some unknown part $x \leq N^{\frac{1}{e}}$. Can we still recover m?

This situation occurs in the case of so-called stereotyped messages: Assume we already know a part M of the message which is always the same, for example M corresponds to "Good morning to everybody. Todays session-key is:". The unknown part x may consist of the words "very secret". One should notice that RSA is frequently used for encrypting session-keys which in turn are then used in symmetric crypto-schemes. But symmetric crypto-schemes often need keys of length at most 80 bits. Hence, the above situation, where the unknown part x is smaller than the e^{th} root of the modulus N can easily occur in practice when RSA is used with small exponent e (for instance with the frequently used choice $e = 3$).

Now, let us apply Coppersmith's method to this problem. An application of Theorem 8 immediately yields the following result.

Theorem 9 (Coppersmith) *Let (N, e) be an RSA public key. Furthermore, let $c := (M + x_0)^e \bmod N$ be an RSA-encrypted message with known M and unknown x_0, where*

$$|x_0| \leq N^{\frac{1}{e}}.$$

Then we can find x_0 in time polynomial in $\log N$ and e.

Proof: Define

$$f_N(x) := (M+x)^e - c,$$

which is an univariate monic polynomial of degree e with the small root x_0, $|x_0| \le N^{\frac{1}{e}}$ modulo N. An application of Theorem 8 proves the claim. $\square$

One should notice that an analogous result holds in the case where the unknown part x is somewhere in the middle of the message, i.e., $m = M + x2^k + M'$ when x begins at the $k+1^{st}$ least significant bit. Observe that we need a small modification in the proof of Theorem 9, since $f_N(x) = (M + x2^k + M')^e - c$ is not a monic polynomial. However the polynomial $2^{-ke} f_N(x) \bmod N$ is monic. Note that the inverse of 2^{ke} modulo N must exist since RSA-moduli are odd. Therefore, we can apply Theorem 8 to the new monic polynomial.

Factoring RSA-moduli $N = pq$ by knowing half of the bits of p

In other scenarios, an attacker might not get parts of the message, but parts of one of the factors p, q of the modulus $N = pq$. For example in the Vanstone-Zuccherato ID-based RSA encryption scheme [69], a person's identity is encoded in his RSA modulus. Vanstone and Zuccherato proposed one variant, where an 1024-bit N is created such that 264 of the most significant bits of p are publically specified.

We will now present an attack — also due to Coppersmith — that finds the factorization of $N = pq$, provided that one knows half of the bits of p. Here we assume wlog that $p > q$. This result breaks the above variant of the Vanstone-Zuccherato scheme.

Here again, we will present the results of Coppersmith in a slightly more general form than originally stated in Coppersmith's work [20], since we will make use of this more general form in the subsequent chapters of the thesis. We obtain Coppersmith's original result in the following theorem for the special case $k = 1$.

Theorem 10 (Coppersmith: MSBs of kp) *Let $N = pq$ with $p > q$. Furthermore, let k be an (unknown) integer that is not a multiple of q. Suppose we know an approximation $\tilde{p}$ of kp with*

$$|\, kp - \tilde{p}\,| \le 2N^{\frac{1}{4}}.$$

Then we can find the factorization of N in time polynomial in $\log N$.

Proof: Define the univariate polynomial

$$f_p(x) := x + \tilde{p},$$

which has the root $x_0 = kp - \tilde{p}$ modulo p with $|x_0| \le 2N^{\frac{1}{4}}$. The polynomial $f_p(x)$ has degree $\delta = 1$ and we know a lower bound of $p \ge N^{\frac{1}{2}}$ since $p > q$. Therefore, we can apply Theorem 7 with the parameter choice β, δ and $c_N = 2$, which gives us the root x_0.

But then we can compute $f(x_0) = kp$. Since k is not a multiple of q, the computation of $\gcd(N, f(x_0))$ yields p. This concludes the proof of the theorem. $\square$

We obtain Coppersmith's original result for the special case $k = 1$ (here we also remove the factor 2).

Theorem 11 (Coppersmith: MSBs of p) *Let $N = pq$ with $p > q$. Suppose we know an approximation $\tilde{p}$ of p with*

$$|\, p - \tilde{p}\,| \leq N^{\frac{1}{4}}$$

Then we can find the factorization of N in time polynomial in $\log N$.

Let us give an interpretation of the result in Theorem 11: We can assume for simplicity that $\tilde{p}$ represents half of the most significant bits of p. Then Coppersmith's method finds the rest of the bits of p in polynomial time and hence the factorization of N.

But what happens if an attacker gets half of the least significant bits of p? Analogous to the case of MSBs of p, one can show similar results when an amount of half of the bits for any intermediate *consecutive* bits are unknown. We will only state the result for the least significant bits of p. A generalization to intermediate consecutive bits is straight-forward, but we consider the case of intermediate bits neither here nor in the subsequent sections of this thesis.

In the following theorem, we assume that we know p modulo some sufficiently large M. In order to interpret this assumption, we can look at the special case $M = 2^i$: This means that we know the i least significant bits of p. However, the formulation of the theorem is more general by allowing arbitrary values of M as long as they are sufficiently large.

Theorem 12 (Coppersmith: LSBs of p) *Let $N = pq$ where p, q are of the same bit-size with $p > q$. Suppose we know p_0 and M satisfying*

$$p_0 = p \bmod M \quad \text{and} \quad M \geq N^{\frac{1}{4}}.$$

Then we can find the factorization of N in time polynomial in $\log N$.

Proof: Define the univariate polynomial

$$f_p(x) := xM + p_0.$$

Since $p_0 = p \bmod M$, the term $x_0 = \frac{p - p_0}{M}$ is an integer. Observe that $f_p(x_0) = p$ and hence x_0 is a root of $f_p(x)$ modulo p. But in order to apply Theorem 7, we need to have a monic polynomial. Therefore, we compute the inverse M^{-1} of M modulo N. If this inverse does not exist then $\gcd(N, M)$ yields the factorization of N.

Next, we compute the monic polynomial

$$f'_p(x) = M^{-1} f_p(x) \bmod N$$

with the same root $x_0 = \frac{p-p_0}{M}$ modulo p. Now, we have to bound the size of $|x_0|$. We know that $q < \sqrt{N} < p < 2\sqrt{N}$, since p and q are of the same bit-size. Using $M \geq N^{\frac{1}{4}}$ yields

$$|x_0| = \frac{p - p_0}{M} \leq 2N^{\frac{1}{4}}.$$

Since $f'_p(x)$ has degree 1 and $p \geq N^{\frac{1}{2}}$, we can apply Theorem 7 with the parameter choice $\delta = 1$, $\beta = \frac{1}{2}$ and $c_N = 2$. We obtain $x = \frac{p-p_0}{M}$ from which we can easily derive p. This concludes the proof of the theorem. $\Box$

Extensions to moduli of the form $N = p^r q$

Recently, there were a number of cryptosystems proposed that rely on the difficulty to factor an modulus of the form $N = p^r q$. For instance, Fujioke, Okamoto and Miyaguchi [30] use a modulus $N = p^2 q$ in an electronic cash scheme. Okamoto and Uchiyama [54] presented a public-key cryptosystem that is provably as secure as factoring a modulus of the form $N = p^2 q$.

In 1998, Takagi [67] observed that the RSA decryption process can be performed significantly faster using moduli of the form $N = p^r q$ for some integer r. Takagi's scheme can be viewed as an extension of the Quisquater-Couvreur method (see Section 2.1): Similarly, one uses the decryption exponents $d_p = d \bmod p - 1$ and $d_q = d \bmod q - 1$ to compute $m^d \bmod p$ and $m^d \bmod q$. The only difference is that one calculates the term $m^d \bmod p^r$ from $m^d \bmod p$ using Hensel lifting before the Chinese Remainder Theorem is applied. The running time of Hensel lifting is negligible compared to the exponentiations. Since p and q can be chosen of smaller bit-size than in the original RSA-scheme, one gains performance.

In 1999, Boneh, Durfee and Howgrave-Graham [16] showed that moduli of the form $N = p^r q$ should be handled with care. Their result generalizes Coppersmith's Theorem (Theorem 10) to moduli of the form $N = p^r q$. In fact, the following theorem is stated in the original work of Boneh, Durfee and Howgrave-Graham for the special case $k = 1$, but we formulate it in a slightly more general way, since we will use this generalization later.

Theorem 13 (BDH) *Let $N = p^r q$, where r is a known constant and p, q are of the same bit-size. Let k be an (unknown) integer that is not a multiple of $p^{r-1} q$. Suppose we know an integer $\tilde{p}$ with*

$$|\, kp - \tilde{p}\,| \leq N^{\frac{r}{(r+1)^2}}.$$

Then N can be factored in polynomial time.

Proof: Define the univariate monic polynomial

$$f_{p^r}(x) := (x + \tilde{p})^r.$$

Then $f(x_0) = (kp)^r = 0 \bmod p^r$ for the root $x_0 = kp - \tilde{p}$ with $|x_0| \le N^{\frac{r}{(r+1)^2}}$.

In order to use Coppersmith's Theorem for monic univariate polynomials (Theorem 7), we have to specify the degree $\delta = r$ of $f_{p^r}(x)$ and a lower bound N^β for the divisor p^r of N. Since $p > \frac{1}{2}q$, we know that $p^{r+1} = \frac{Np}{q} > \frac{1}{2}N$. This implies

$$p^r > \left(\frac{1}{2}N\right)^{\frac{r}{r+1}} > \frac{1}{2}N^{\frac{r}{r+1}}$$

Hence we can define $\beta := \frac{r}{r+1} - \frac{1}{\log N}$. An application of Theorem 7 with the parameter choice δ, β and $c_N = 2$ shows that we can find all roots that are of absolute value at most

$$2N^{\frac{\beta^2}{\delta}} = 2N^{\frac{r}{(r+1)^2} - \frac{2}{(r+1)\log N} + \frac{1}{r \log^2 N}} \ge 2N^{\frac{r}{(r+1)^2} - \frac{1}{\log N}} = N^{\frac{r}{(r+1)^2}}$$

Thus, we can find the value $x_0 = kp - \tilde{p}$ which then in turn gives us the value of kp.

Since k is not a multiple of $p^{r-1}q$, we know that kp is not a multiple of N. Therefore $\gcd(N, kp)$ is either of the form p^i or of the form qp^j for some integers $i \le r$, $j < r$. If we obtain p^i then we can find p by guessing i and computing the i^{th} root of p^i. In the case qp^j, we compute $\frac{N}{qp^j}$ and further proceed as in the first case. Hence, we obtain the complete factorization of N which concludes the proof of the theorem. □

Let us briefly discuss the implications of Theorem 13 in terms of the fraction of bits of p that is sufficient to factor N, i.e., we only consider the special case where $k = 1$. Since N is of size roughly p^{r+1}, one could also formulate the theorem for an approximation $\tilde{p}$ with

$$|\, p - \tilde{p} \,| \le p^{\frac{r}{r+1}}.$$

Thus, we need an amount of roughly $\frac{1}{r+1}$ of the bits of p in order to factor N. That means for the case $r = 1$, we need half of the most significant bits of p, which is exactly Coppersmith's result. So for example if N is an 1000-bit modulus then we need about 250 bits of p. But for $r = 2$, we need only one third of the bits of p. Again, if N is 1000 bit then p, q are of size 333 bit and we only need to know 111 bit.

Boneh, Durfee and Howgrave-Graham remarked that for

$$r = \Omega\left(\frac{\log N}{\log \log N}\right),$$

a fraction of $\mathcal{O}(\frac{\log \log N}{\log N})$ of the bits of p is sufficient. But this is a total amount of $\mathcal{O}(\log \log N)$ bits, which in turn can be guessed in time $\mathcal{O}(\log N)$. Therefore, moduli of the form $N = p^r q$ can be factored in time polynomial in the bit-length of N if $r = \Omega(\frac{\log N}{\log \log N})$. However in cryptographic applications, r is normally a small constant.

3.4 The multivariate case

"When the mathematician says that such and such a proposition is true of one thing, it may be interesting, and it is surely safe. But when he tries to extend his proposition to everything, though it is much more interesting, it is also much more dangerous."
E. Kasner and J. Newman

A very natural question is whether one can extend Coppersmith's to the case of multivariate modular polynomial equations. Hence, we will study here the following problem:

> Let N be a composite number of unknown factorization and b be a divisor of N, where $b \geq N^{\beta}$ for some known β. Given a polynomial $f_b(x_1, \ldots, x_k) \in \mathbb{Z}[x_1, \ldots, x_k]$ with degree δ_i in each variable x_i. Find in time polynomial in $(\log N, \max_i\{\delta_i\}, k)$ all roots $(r_1, \ldots, r_k) \in \mathbb{Z}^k$ of $f_b(x_1, \ldots, x_k)$ modulo b, where $|r_i| < X_i$ for some bounds X_i, $i = 1, \ldots, k$.

In principle there is no problem in applying Coppersmith's methodology from the previous section that transforms modular polynomials into integer polynomials with the same small root. That is, we can construct from $f_b(x_1, \ldots, x_r)$ a polynomial $f(x_1, \ldots, x_r)$ with the same small root over the integers and not just modulo b.

We can explicitly state a condition on the upper bounds $X_1, \ldots, X_k$ which tells us when this reduction from the modular to the integer case succeeds. This condition is given in the following Lemma due to Howgrave-Graham which is a direct generalization of Theorem 5 to the multivariate case.

Theorem 14 (Howgrave-Graham) *Let $f(x_1, \ldots, x_k)$ be a polynomial in k variables with n monomials. Further, let m be a positive integer. Suppose that*

(1) $f(r_1, \ldots, r_k) = 0 \bmod b^m$ *where* $|r_i| < X_i$, $i = 1, \ldots, k$

(2) $\|f(x_1X_1, \ldots, x_kX_k)\| < \frac{b^m}{\sqrt{n}}$

Then $f(r_1, \ldots, r_k) = 0$ holds over the integers.

The proof of Theorem 14 is completely analogous to the proof of Theorem 5 in the previous section. Therefore, we omit it. We will use Theorem 14 frequently in the subsequent chapters of this thesis in the bivariate and trivariate case.

Up to now, everything works completely similar to the univariate modular case. But assume that we manage to construct a polynomial $f(x_1, \ldots, x_k)$ with a small root over the integers, then there remains a serious problem:

Problem: How can we extract the integer roots of $f(x_1, x_2, \ldots, x_k)$?

Whereas root finding for univariate polynomials can be done in time polynomial in the input-size, one cannot solve the problem for multivariate polynomials in general. A simple reason for the impossibility of an efficient solution is the number of possible solutions. By a fundamental theorem of algebra, the number of integer solutions of an univariate polynomial $f(x)$ is bounded by its degree. Now consider for instance the bivariate polynomial

$$f_1(x, y) := x - y.$$

This polynomial has infinitely many integer solutions which all lie on the line $y = x$. Therefore, one cannot enumerate the solutions.

Let us consider a second bivariate polynomial $f_2(x, y) := x + y - 2$. The integer roots of this polynomial lie on the line $y = 2 - x$, which crosses the line $y = x$ in the point $(x_0, y_0) = (1, 1)$. Thus, the point (x_0, y_0) is a common zero of both polynomials. Can we find common zeros of bivariate polynomials in general? Then one could generalize Coppersmith's method in the following way:

> Instead of finding just one polynomial $f(x_1, \ldots, x_k)$ that has some small root $(r_1, \ldots, r_k)$ over the integers, we use the L^3-algorithm in order two find k different polynomials with the same small root over the integers. Then we compute the common roots of all polynomials.

Unfortunately, this approach cannot work in general. Let us consider the polynomial $f_3(x, y) := x^2 - y^2$. Since $\gcd(f_1, f_3) = f_1$, the common zeros of f_1 and f_3 are the roots of the polynomial f_1. Hence it seems that we did not gain very much. However, it might be possible to extract common roots if the number of common roots is not too large.

The resultant heuristic

Let us look at projections of the common roots on one variable. Define

$$P_x := \{x \in \mathbb{Z} \mid \text{ There is an } y \in \mathbb{Z} \text{ such that } f_1(x, y) = f_2(x, y) = 0\}.$$

Then P_x is the projection of the common roots of f_1 and f_2 onto the x-coordinate. It remains to show that we can efficiently compute the set P_x. This can be done using resultants of polynomials (for an introduction into the theory of resultants see [22, 23, 72]).

Let $r(x) := \mathrm{res}_y(f_1, f_2)$ be the resultant of f_1 and f_2 with respect to the variable y. Then $r(x)$ is an univariate polynomial in the variable x, which can be efficiently computed as the determinant of a Sylvester matrix that consists of shifted versions of f_1 and f_2 (for details see [22]).

Moreover, let

$$R_x := \{x \in \mathbb{Z} \mid r(x) = 0\}.$$

It is a well-known fact that $P_x \subseteq R_x$ (see for instance [23]). Therefore, we can compute all elements of P_x by computing the integer roots of the polynomial $r(x)$. Symmetrically, we can compute $\mathrm{res}_x(f_1, f_2)$ in order to find the candidates for the y-coordinates of the common roots.

Let us consider the resultant computations for our simple examples $f_1(x, y) := x - y$, $f_2(x, y) := x + y - 2$ and $f_3(x, y) := x^2 - y^2$. We compute $\mathrm{res}_y(f_1, f_2) = x - 1$ and $res_x(f_1, f_2) = y - 1$, which gives us the unique common root $(1, 1)$ of both polynomials. On the other hand, we obtain $\mathrm{res}_y(f_1, f_3) = \mathrm{res}_x(f_1, f_3) = 0$. Since the resultant is an univariate polynomial of finite degree, it contains infinitely many zeros iff it is the zero polynomial. That means: Whenever two polynomials have infinitely many roots in common (these common roots must only be roots over the complex numbers $\mathbb{C}$) then the resultant of these polynomials is identical to the zero polynomial. In this case, f_1 and f_2 must share a nontrivial greatest common divisor.

Hence, we can compute the common roots of two bivariate polynomials if their resultant is not identical to the zero polynomial. One can extend this approach to k-variate polynomials. Let $f_1, \ldots, f_k$ be polynomials in the variables $x_1, \ldots, x_k$. Then the $k - 1$ resultants

$$h_1 = \mathrm{res}_{x_1}(f_1, f_2), h_2 = \mathrm{res}_{x_1}(f_2, f_3), \ldots, h_{k-1} = \mathrm{res}_{x_1}(f_{k-1}, f_k)$$

are $(k-1)$-variate polynomials in the variables $x_2, \ldots, x_k$. Thus, we have eliminated the variable x_1. If none of these resultants is the zero polynomial, we can further eliminate the variable x_2 by computing the resultants

$$\mathrm{res}_{x_2} = (h_1, h_2), \ldots, \mathrm{res}_{x_2}(h_{k-2}, h_{k-1}).$$

We can proceed in this way until we have eliminated all but the last variable x_k. Then the last resultant is an univariate polynomial in x_k which can be solved by standard root finding algorithms.

Hence in order to generalize Coppersmith's method to the multivariate modular case, we construct k different k-variate polynomials $f_1, \ldots, f_k$ with some common small root. The construction of these polynomials is analogous to the univariate case. Afterwards, we try to extract the common roots by resultant computations. This procedure fails if in some step one of the resultants is the zero polynomial.

Coppersmith Method in the multivariate case (informally)

INPUT: – Polynomial $f_b(x_1, x_2, \ldots, x_k)$ of degree δ_i in each variable x_i, $i = 1, \ldots, k$.

– Modulus N of unknown factorization which is a multiple of b and a lower bound $b \geq N^{\beta}$

Step 1: Fix an integer m (depending on N, β, δ_i and k) and construct a set G of k-variate polynomials, where each $g_i(x_1, \ldots, x_k) \in G$ satisfies

$$g(x_1, \ldots, x_k) = 0 \bmod b^m.$$

Step 2: Construct the lattice L from the $g_i(x) \in G$: The basis vectors of L are the coefficient vectors of $g_i(x_1X_1, \ldots, x_kX_k)$, where the bounds X_i, $i = 1, \ldots, k$ are chosen such that the L^3-algorithm finds k vectors with norm smaller than $\frac{b^m}{\dim(L)}$ in L. The size of each bound X_i is a function of the parameters N, β, δ_i and k.

Step 3: Apply the L^3-algorithm to the lattice bases. Let $v_1, \ldots, v_k$ be the k shortest vectors in the L^3-reduced bases. The vectors $v_1, \ldots, v_k$ are the coefficient vectors of some polynomial $f_1(x_1X_k, \ldots, x_kX_k), \ldots, f_k(x_1X_1, \ldots, x_kX_k)$. Construct $f_1(x_1, \ldots, x_k), \ldots, f_k(x_1, \ldots, x_k)$ from $v_1, \ldots, v_k$.

Step 4: For $i = 1, \ldots, k$:
Compute via resultant computations an univariate polynomial $h_i(x_i)$ from $f_1, \ldots, f_k$.

OUTPUT: Polynomials $h_i(x_i)$ with $h_i(r_i) = 0$ over $\mathbb{Z}$ whenever $f_b(r_1, \ldots, r_k) = 0 \bmod b$ and $|r_i| \leq X_i$ for $i = 1, \ldots, k$.

The method fails to find the roots if the $h_i(x_i)$ are zero polynomials. The problem is that the k shortest vectors of an L^3-reduced basis are linearly independent, but we cannot guarantee that the corresponding polynomials that are constructed from these vectors are independent in an algebraical sense: For instance, one polynomial may be a non-trivial multiple of another polynomial. In this case, at least one resultant computation fails, and therefore the whole approach must fail.

However, one should not overemphasize the negative aspects of the multivariate method. We will frequently use this method throughout the thesis. Our experiments confirm that the resultant computations are in many situations a useful method in order to extract roots of multivariate polynomials over the integers. Moreover, we made the following interesting experimental observation:

> If we had a fixed construction rule how to choose the set G of polynomials then either the resultants always yielded the desired small root $r_1, \dots, r_k$ or the resultants were always identically zero. That means for all the methods described in this thesis, the resultant heuristic either succeeded in every experiment or the resultant heuristic always failed.

We give several applications of Coppersmith's multivariate method in this thesis using resultant computations. Since extracting roots by resultant computations in Coppersmith's method is a heuristic, we will refer to this method as the *resultant heuristic*. In each case, we tested this heuristic by several experiments, which show that the heuristic is very useful in practice. Since the resultant heuristic is the only heuristical part in Coppersmith's multivariate approach, we will often make the assumption that the heuristic always works such that we can state our results as theorems. However, in each case where we assume that the resultant heuristic succeeds, we make this assumption explicit.

It would be very nice to formulate many of the results in the subsequent chapters without relying on the resultant heuristic. Therefore, we consider the following problem as one of the most important theoretical problems arising from the results in this thesis:

Open problem: Find explicit conditions under which Coppersmith's method in the multivariate case succeeds to find small roots.

As we mentioned before, we encountered some sets G of polynomials where the resultant heuristic always fails. Interestingly, those were mostly special cases where we could find rigorous methods for the multivariate modular approach. Hence a failure of the resultant heuristic must not imply a failure of Coppersmith's multivariate approach in general! Moreover, for polynomials f_b of a special form there might be alternative ways to find the desired small roots. Hence, one of our goals throughout this thesis will be:

Goal: Find special cases where Coppersmith's multivariate method is provable.

4 Weak Keys in RSA

"The trouble with the integers is that we have examined only the small ones."
Ronald Graham

4.1 Introduction

All the attacks that are proposed in this chapter as well as in the subsequent chapters are factorization algorithms. Since factoring an RSA modulus is assumed to be a hard problem and since our goal is to find polynomial time attacks on RSA, we have to relax the problem of factoring N. That means, we have to indentify practically interesting special cases, where the factorization problem is solvable in polynomial time.

In this chapter, the main idea to make the factorization problem feasible is to use the information given by the RSA public exponent e. Hence, the central question we will study here is:

Question: When does e provide enough information to factor N?

At first sight, this question seems to make not much sense because there is no reason that e should give an attacker any useful information at all about the factorization. The only thing we learn is that e is in $\mathbb{Z}^*_{\phi(N)}$. Indeed most of the known cryptanalytic attacks on RSA focus on the difficulty to factor the modulus N without taking into account additional information that might be encoded in the public exponent e. Hence it is tempting for crypto-designers to use public exponents e of a very special structure that yield good performance in the encrypting/decryption process. For example, one might be tempted to use public exponents e that correspond to small secret exponents d in order to speed up the decryption process.

Another possibility of an RSA-variant with a fast decryption process was introduced by Yen, Kim, Lim and Moon [73, 74] in 2001. This YKLM-scheme is designed to counteract the fault-based attack on CRT-RSA (i.e., RSA with Chinese Remaindering) of Boneh, DeMillo and Lipton [11]. In order to be efficient, the YKLM-scheme uses a new key generation process that produces public keys (N, e) of a special form.

In 1990, Wiener [71] was the first one who observed that information encoded in the public exponent e might help to factor the modulus N. He showed that every public exponent e that corresponds to a secret exponent $d \leq \frac{1}{3}N^{\frac{1}{4}}$ yields the factorization of N in time polynomial in the bit-size of N. This result is up to now the most famous attack on RSA. In order to recover the secret key and the factorization of N, Wiener applied the continued fraction method — a variant of the Euclidean Algorithm — to the public key tuple (N, e). We prove Wiener's result in Section 4.2 using Coppersmith's method (see Chapter 3) in combination with two-dimensional lattices.

In 1999, Boneh and Durfee [12] improved upon Wiener's result by showing that every public exponent e that corresponds to a secret exponent $d \leq N^{0.292}$ yields the factorization of N (see Section 7.2). The Boneh-Durfee attack makes use of Coppersmith's method for finding roots of bivariate modular polynomial equations (see Section 3.4). As opposed to Wiener's method, this approach is heuristic. However, many experiments by various researchers confirm that this heuristic works very well in practice. In fact, no systematic failure is known and the method is supposed to find the factorization of N when applied with suitable parameter choices.

The Wiener attack as well as the Boneh-Durfee attack cannot be applied to the YKLM-scheme [73, 74]. Although the YKLM-scheme uses a special key generation algorithm in order to provide good decryption performance, the secret keys d are not chosen to be small. On the other hand, in Section 4.3 we present an extension of Wiener's approach that leads to a much larger class of secret keys d which are insecure. Furthermore, in Section 4.4 we show that the keys which are generated in the YKLM-scheme belong to this larger class, for all reasonable parameter choices of the scheme. As a result, we obtain that the public keys (N, e) in the YKLM-scheme also yield the factorization of N in polynomial time.

Let us put the cryptanalytic approaches above into a more general framework by defining the notion of *weak keys*: The results so far show that there are classes of public keys (N, e), where every element in the class yields the factorization of N. For instance, in the case of the Wiener attack the class consists of all public key tuples (N, e) where $ed - 1 = 0 \bmod \phi(N)$ with $d < \frac{1}{3}N^{\frac{1}{4}}$. One may view the auxiliary input e as a hint how to factor N: Without having e we assume that factoring N is hard, but with the help of e it becomes feasible.

We call a class in which every element (N, e) yields the factorization of N *weak* and the elements (N, e) of the weak class are called *weak keys*. To be more precise:

Definition 15 *Let C be a class of RSA public keys (N, e). The size of the class C is defined by*

$$size_C(N) := |\{e \in \mathbb{Z}^*_{\phi(N)} \mid (N, e) \in C\}|.$$

C is called weak *if:*

1. *$size_C(N) = \Omega(N^\gamma)$ for some $\gamma > 0$.*
2. *There exists a probabilistic algorithm A that on every input $(N, e) \in C$ outputs the factorization of N in time polynomial in* $\log(N)$.

The elements of a weak class are called weak keys.

Note that the size of a weak class is a function in N which denotes the number of elements that can be factored by the corresponding algorithm A. For example, the size of the class in the Wiener attack is at least $N^{\frac{1}{4}-\epsilon}$. Here the ϵ-term comes from the fact that only those d with $\gcd(d, \phi(N)) = 1$ define legitimate RSA keys.

Let us give another (trivial) example of a weak class of public keys. Every tuple (N, e) with $e = kq$, $1 < k < p$ is a weak key, since the computation $\gcd(N, e) = q$ yields the factorization. These are $p > N^{\frac{1}{2}}$ many weak keys (remember that we assume wlog that $p > q$). By Theorem 10, we see that even the knowledge of $e = kq + r$ for some unknown $|r| \leq N^{\frac{1}{4}}$ suffices to find the factorization of N. This implies the existence of a weak class with size $\Omega(N^{\frac{3}{4}})$.

We think that it is a very natural question to study how many of the possible choices of the public keys are indeed weak keys that should not be used in the design of cryptosystems. For the Wiener attack and the Boneh-Durfee attack it is easy for a crypto-designer to see that a key is weak by inspecting the most significant bits of d. For the extension of Wiener's attack that we describe in Section 4.3, the weakness of the keys is not obvious. One can understand our new result as a warning for crypto-designers to be careful when using keys with a special structure.

There also is an imminent danger from weak keys in the case of untrusted servers that create public/secret key pairs: Crépeau and Slakmon [25] showed how to use weak keys in order to construct malicious RSA systems by encoding information into the public exponent e. Our new class of weak keys is well-suited for the use in such systems and leads to a large variety of new malicious keys.

Thus, we can define the main goal of this chapter as following:

Goal: Identify weak classes with size as large as possible.

Moreover, we demand that our weak classes have some practical applications in the design of RSA-based cryptosystems. For instance, the weakness of the class of keys (N, e) with $e = kq + r, |r| \leq N^{\frac{1}{4}}$ that was introduced in the example above, shows that one variant of the Vanstone-Zuccherato ID-based RSA encryption scheme [69] can be broken in polynomial time (this was already noticed by Coppersmith [20]).

We show that the following practically interesting class C is weak. The class C contains all the keys (N, e) where e satisfies a relation

$$ew + z = 0 \bmod \phi(N)$$

for some small, unknown parameters w and z. In order to provide bounds for the size of w and z, let us first consider the normal RSA-case, where $p - q = \Omega(\sqrt{N})$. Note, that for randomly chosen primes of the same bit-size, the probability that p and q agree in the c most significant bits is roughly $2^{-(c-1)}$. Hence, we have $p - q = \Omega(\sqrt{N})$ with overwhelming probability.

For the case $p - q = \Omega(\sqrt{N})$, we introduce a variant of Wiener's attack that works for all public keys (N, e) where $ew + z = k\phi(N)$, $k \in \mathbb{N}$ with

$$w \leq \frac{1}{3} N^{\frac{1}{4}} \quad \text{and} \quad |z| = \mathcal{O}(N^{-\frac{3}{4}} ew).$$

We want to point out that our parameter choices in the attack exclude trivial solutions $ew + z = 0$ over the integers, since $|z| < ew$. Thus, our parameters always guarantee that $ew + z$ is a positive multiple of $\phi(N)$.

Since we want to show that the above class of keys is indeed weak, we have to prove the existence of a probabilistic polynomial time algorithm that on input (N, e) outputs the factors p and q. For our class, we present an explicit factorization algorithms that finds p and q. Thus, our method is constructive: Given an RSA key tuple (N, e), one can run our algorithm to test whether the key belongs to the corresponding weak class or not. This property may be useful in the design of cryptosystems during the key generation process to ensure that one does not accidentally create a weak key (although the probability is negligible if e is chosen randomly in $\mathbb{Z}^*_{\phi(N)}$).

Our result can be seen as a generalization of Wiener's result. In Wiener's attack, the weak keys (N, e) have the special structure that $ed - 1 = 0 \bmod \phi(N)$ for some secret exponent $d \leq \frac{1}{3} N^{\frac{1}{4}}$. Hence Wiener's method is a special case of our attack, where $w = d$ and $z = -1$. If we want to compare Wiener's results to ours in terms of the secret key d, then Wiener's method works whenever $d \leq \frac{1}{3} N^{\frac{1}{4}}$ whereas our method applies to all secret keys of the form

$$d = -\frac{w}{z} \bmod \phi(N) \quad \text{with } w \leq \frac{1}{3} N^{\frac{1}{4}} \text{ and } |z| = \mathcal{O}(N^{-\frac{3}{4}} ew).$$

It is important to notice that in contrast to the approaches of Wiener and Boneh-Durfee, the secret keys in our attack are not small itself but have a "small decomposition" in w and z. So they might look innocuous to crypto-designers and may be tempting to use in the design of cryptosystems with good encryption/decryption performance.

As an example, in Section 4.4 we show that the public keys (N, e) constructed in the YKLM-scheme can be attacked by our generalization of Wiener's method. Namely, we can express the secret exponent d in terms of small w and z, which breaks the cryptosystem for all reasonable parameter choices.

Similar to Wiener's method (Section 4.2), in our generalization we can use a two-dimensional lattice to recover the unknown parameter w. Then, in order to find the unknown parameter z and the factorization, our approach is based on Coppersmith's method that factors N given the upper half of the bits of p (see Theorem 11).

In this chapter our goal is to identify weak classes of maximal size. But how many weak keys (N, e) do we have in the case $ew - z = 0 \bmod \phi(N)$ with

$$w \leq \frac{1}{3} N^{\frac{1}{4}} \quad \text{and} \quad |z| = \mathcal{O}(N^{-\frac{3}{4}} ew)?$$

One should observe that for w of size roughly $N^{\frac{1}{4}}$, the parameter e must be of size at least $N^{\frac{3}{4}}$ in order to satisfy a relation of the form $ew + z = 0 \bmod \phi(N)$. Thus, $|z|$ can be chosen of size at least w. If e is roughly N, which is normally the case for small d, then in the attack $|z|$ can even be chosen of size $N^{\frac{1}{4}}w$.

One should expect that for fixed N the number of public keys (N, e) for which our approach applies is roughly the number of tuples (w, z) within the given bounds. By the observation above, this number can be upper bounded by $w \cdot N^{\frac{1}{4}}w \leq N^{\frac{3}{4}}$. Interestingly, we are able to show that the size of the class C of keys (N, e) for which our algorithm works is also lower bounded by

$$\text{size}_C(N) = \Omega(N^{\frac{3}{4}-\epsilon}).$$

Furthermore, this bound can be improved if we skip the restriction that $p - q = \Omega(\sqrt{N})$. In 2001, de Weger [70] generalized Wiener's method to arbitrary prime differences

$$p - q = N^{\frac{1}{4}+\gamma}\ , \quad \text{where } 0 \leq \gamma \leq \frac{1}{4}.$$

It is important to notice that for prime differences $p-q = \mathcal{O}(N^{\frac{1}{4}})$ an algorithm of Fermat finds the factorization in polynomial time (see [40], Section V.3).

de Weger showed that Wiener's method succeeds whenever $d \leq N^{\frac{1}{2}-\gamma}$. In terms of weak keys (N, e) this means that all those keys (N, e) are weak where $e \in \mathbb{Z}^*_{\phi(N)}$ with corresponding secret exponent $d \leq N^{\frac{1}{2}-\gamma}$. Therefore, the size of de Weger's weak class is $\Omega(N^{\frac{1}{2}-\gamma-\epsilon})$.

de Weger's method also applies to our extension of Wiener's attack. Interestingly, we are able to show that for prime differences $p - q = N^{\frac{1}{4}+\gamma}$, $0 < \gamma \leq \frac{1}{4}$ our attack defines a weak class C of size

$$\text{size}_C(N) = \Omega(N^{1-\gamma-\epsilon}).$$

Thus, our attack has a nice interpolation property towards Fermat's algorithm: As $p-q$ decreases, the number of weak public keys increases. For γ approaching zero almost all keys are weak, corresponding to the fact that N can be easily factored without any hint that is encoded in e.

As a by-product, we get a simple probabilistic factorization algorithm with expected running time $\mathcal{O}(N^{\gamma+\epsilon})$ comparable to Fermat-Factorization: For a fixed N, choose random $e < N$ and apply our algorithm to each choice (N, e) until (N, e) is a weak key that yields the factorization.

Notice that the interpolation property above seems to imply that one cannot improve our approach significantly. On the other hand, there might be different techniques – for example lattice reduction techniques for higher dimensional lattices – that lead to larger classes of weak keys for the prime difference $p - q = \Omega(\sqrt{N})$. But at the moment this is an open question.

4.2 Wiener's attack

"Although this may seem a paradox, all exact science is dominated by the idea of approximation."
Bertrand Russell

We recall Wiener's famous attack on RSA with secret exponent $d \leq \frac{1}{3}N^{\frac{1}{4}}$. As opposed to Wiener, we do not state the results in terms of continued fractions but in terms of lattices. Namely, we use a two-dimensional lattice. Our approach has three main advantages over Wiener's method:

- In order to find the secret key d with the help of continued fractions, one has to test among $\mathcal{O}(\log N)$ candidates in the continued fraction expansion of $\frac{e}{N}$, whereas in our method we get d directly.
- We are able to prove a generalization of Wiener's method due to Verheul and Tilborg [68] in a much simpler fashion than stated in the original paper.
- The two-dimensional lattice approach will be used again in Chapter 5, where we introduce another attack on RSA with so-called small CRT-exponents. One can improve this attack by generalizing to arbitrary lattice dimension (see Section 5.3), which is not possible using continued fractions.

Theorem 16 (Wiener) *Let (N, e) be an RSA public key with $e \in \mathbb{Z}^*_{\phi(N)}$ and corresponding secret exponent d. Suppose*

$$d \leq \frac{1}{3}N^{\frac{1}{4}}.$$

Then N can be factored in time $\mathcal{O}(\log^2(N))$.

Proof. Let us start by looking at the RSA key-equation

$$ed = 1 \bmod \phi(N).$$

This equation can also be written as $ed = 1 + k\phi(N)$ for some integer k. Note that

$$k = \frac{ed-1}{\phi(N)} < \frac{e}{\phi(N)}d < d.$$

Since $\phi(N) = (p-1)(q-1) = N - p - q + 1$, we can rewrite our equation as

$$ed + k(p + q - 1) - 1 = kN. \tag{4.1}$$

This gives us a bivariate polynomial

$$f_N(x, y) = ex + y$$

with a root $(x_0, y_0) = (d, k(p + q - 1) - 1)$ modulo N.

Roadmap of the proof

- Applying Coppersmith's method, we transform the polynomial $f_N(x, y)$ into a polynomial $f(x, y)$ which has the root (x_0, y_0) over the integers and not just modulo N.
- We show how to extract the root (x_0, y_0) from the polynomial $f(x, y)$.
- The term (x_0, y_0) gives us the factorization of N.

In order to apply Coppersmith's method, we use a special case of Howgrave-Graham's theorem (Theorem 14) for the bivariate case.

Theorem 17 (Howgrave-Graham) *Let $f(x, y)$ be a polynomial that is a sum of at most two monomials. Suppose*

(1) $f(x_0, y_0) = 0 \bmod N$, *where* $|x_0| \leq X$ *and* $|y_0| \leq Y$

(2) $\|f(xX, yY)\| < \frac{1}{\sqrt{2}}N$

Then $f(x_0, y_0) = 0$ *holds over the integers.*

In order to find upper bounds for x_0 and y_0, we define $X := \frac{1}{3}N^{\frac{1}{4}}$. Since $k < d$ and $p + q \leq \frac{3}{\sqrt{2}}\sqrt{N}$ (see Section 2.1), we obtain

$$y_0 \leq \frac{3}{\sqrt{2}}\sqrt{N}X.$$

We define $Y := \frac{3}{\sqrt{2}}\sqrt{N}X$. Thus, we have $x_0 \leq X$ and $y_0 \leq Y$.

With the definition of X and Y, our polynomial f_N satisfies condition *(1)* of Theorem 17. Notice that

$$\|f_N(xX, yY)\| = \sqrt{(eX)^2 + Y^2} > eX \geq ed > k\phi(N) > N.$$

Therefore f_N does not satisfy condition *(2)*.

In order to construct a polynomial f which satisfies both conditions, we use the auxiliary polynomial $f_0(x, y) = Nx$. This polynomial trivially satisfies condition *(1)*, since f_0 is the zero polynomial modulo N. Thus every integer linear combination

$$g(x, y) = c_0 f_N(x, y) + c_1 f_0(x, y) \qquad c_1, c_2 \in \mathbb{Z}$$

satisfies condition *(1)*. We have to search for a small norm integer linear combination that also satisfies condition *(2)*. The coefficient vectors of $g(xX, yY)$ form a lattice L in

$\mathbb{Z}^2$. L is spanned by the coefficient vectors of $f_0(xX, yY)$ and $f_N(xX, yY)$. Therefore, a basis B of L is given by the span of the row vectors of the following (2×2)-matrix:

$$B = \begin{bmatrix} NX & \\ eX & Y \end{bmatrix}.$$

In order to satisfy condition *(2)*, we have to find a vector $v = (c_0, c_1) \cdot B \in L$ with $\|v\| < \frac{1}{\sqrt{2}}N$. Such a vector v is the coefficient vector of a bivariate polynomial $f(xX, yY) = (c_0 N + c_1 e)Xx + c_1 Yy$. Thus, the polynomial $f(x, y)$ satisfies both conditions in Theorem 17.

Note that a shortest vector in L can be found by using the Gauss reduction algorithm (see [58]). Therefore, we have to prove that L indeed contains a sufficiently short vector.

Lemma 18 *L contains a shortest vector with norm smaller than $\frac{1}{\sqrt{2}}N$.*

Proof: By Minkowski's theorem (Theorem 3), L must contain a vector v with $\|v\| \leq \sqrt{2\det(L)}$. Hence $\|v\| < \frac{1}{\sqrt{2}}N$, if the condition

$$2\det(L) < \frac{1}{2}N^2$$

holds. Since $\det(L) = NXY = \frac{3}{\sqrt{2}}N^{\frac{3}{2}}X^2$, we obtain

$$X^2 < \frac{\sqrt{2}}{12}N^{\frac{1}{2}}.$$

But $X = \frac{1}{3}N^{\frac{1}{4}}$, which proves the claim. □

By Lemma 18, our lattice L contains a shortest vector $v = (c_0, c_1) \cdot B$ with $\|v\| < \frac{1}{\sqrt{2}}N$. This vector v corresponds to a bivariate polynomial $f(x, y) = (c_0 N + c_1 e)x + c_1 y$. We know by Theorem 17 that $f(x, y)$ has the root (x_0, y_0) over the integers. The following lemma shows that we obtain the unknown root (x_0, y_0) directly from f's coefficients $(c_0 N + c_1 e)$ and c_1.

Lemma 19 *Let $v = (c_0, c_1) \cdot B$ be a shortest vector in L. Then*

$$(x_0, y_0) = (|c_1|, |c_0 N + c_1 e|).$$

Proof: By Lemma 18 and Theorem 17, we have

$$f(x_0, y_0) = (c_0 N + c_1 e)x_0 + c_1 y_0 = 0.$$

This implies

$$\frac{x_0}{y_0} = -\frac{c_1}{c_0 N + c_1 e}$$

The goal is to show that the fractions on both sides of the equality are in their lowest terms. This would imply that the nominators and denominators are equal up to the sign. But x_0 and y_0 are positive integers by construction which proves the lemma.

Our first claim is that $\gcd(x_0, y_0) = \gcd(d, k(p+q-1)-1) = 1$. By equation (4.1), we know that $\gcd(d, k(p+q-1)-1)$ divides the term kN. Therefore, it suffices to show that $\gcd(d, kN) = 1$. From the equation $ed - k\phi(N) = 1$, we see that $\gcd(d, k) = 1$. But we also know that $d < \frac{1}{3}N^{\frac{1}{4}}$, which implies $\gcd(d, p) = 1$ as well as $\gcd(d, q) = 1$. This concludes the proof of the first claim.

Our second claim is that $\gcd(c_1, c_0 N + c_1 e) = 1$. First, note that $\gcd(c_1, c_0 N + c_1 e) = \gcd(c_1, c_0 N)$. Our vector v is a shortest vector in L. Hence, we must have $\gcd(c_1, c_0) = 1$, since otherwise we could divide v by $\gcd(c_1, c_0)$, obtaining a shorter vector. This implies $\gcd(c_1, c_0 N) = (c_1, N)$. Now observe that Lemma 18 implies that $c_1 Y < \frac{1}{\sqrt{2}}N$. Thus, $c_1 < \frac{N}{\sqrt{2}Y} = \frac{\sqrt{N}}{3X} = N^{\frac{1}{4}}$. Hence we get $\gcd(c_1, p) = 1$ as well as $\gcd(c_1, q) = 1$. This concludes the proof of the second claim and of the lemma. □

By Lemma 19, we obtain the tuple (x_0, y_0). Now, we can use equation (4.1) to compute

$$k = \frac{ex_0 + y_0}{N}.$$

Using equation (4.1) again, we see that

$$p + q = \frac{1 - ed}{k} + N + 1.$$

If we substitute $q = \frac{N}{p}$ and multiply both sides by p, we finally obtain

$$p^2 - \left(\frac{1 - ed}{k} + N + 1\right) p + N = 0.$$

This is a quadratic equation over the integers in the unknown parameter p. It can easily be solved in time polynomial in $\log N$ and its two solutions are the desired factors p, q of N.

Let us briefly summarize the whole factorization algorithm.

Algorithm Wiener-Attack

INPUT: (N, e), where $N = pq$

1. Construct the lattice L with basis B.
2. Find a shortest vector $v = (c_0, c_1) \cdot B \in L$ using Gauss reduction.
3. Compute $(x_0, y_0) = (|c_1|, |c_0 N + c_1 e|)$ and $k = \frac{ex_0 + y_0}{N}$.
4. Output the two solutions of the quadratic equation

$$z^2 - \left(\frac{1 - ex_0}{k} + N + 1 \right) z + N = 0.$$

OUTPUT: p, q

Since every step in the Algorithm Wiener-attack can be done in time $\mathcal{O}(\log^2(N))$, this concludes the proof of Theorem 16. $\diamond$

The result of Verheul and Tilborg

Let us recall that for Wiener's result, the RSA equation $ed - 1 = k(N - (p + q - 1))$ gives us a polynomial

$$f_N(x, y) = ex - y \text{ with root } (x_0, y_0) = (d, k(p + q) + 1) \bmod N.$$

Now, Verheul and Tilborg [68] studied the case where an attacker additionally guesses high order bits of p. Assume we know $\tilde{p}$ with $|p - \tilde{p}| \leq N^{\frac{1}{2} - \gamma}$ and by calculating $\tilde{q} = \frac{N}{\tilde{p}}$ we know an approximation of q with accuracy $N^{\frac{1}{2} - \gamma}$ as well.

In this case, the RSA equation $ed - 1 = k(N - (p + q - 1))$ gives us a polynomial

$$f_{N'}(x, y) = ex - y \text{ with root } (x_0, y_0') = (d, k(p - \tilde{p} + q - \tilde{q}) + 1) \bmod N'$$

where $N' = N + 1 - \tilde{p} - \tilde{q}$.

We have $|y_0'| \leq dN^{\frac{1}{2} - \gamma}$. Working through the arithmetic of the proof of Theorem 16 and neglecting constants, this gives us the condition

$$d \leq N^{\frac{1}{4} + \frac{\gamma}{2}}.$$

Wiener's result follows as the special case where $\gamma = 0$, i.e., where the attacker guesses no additional bits.

Let us interpret the result of Verheul and Tilborg: In order to improve Wiener's bound by r bits, an attacker has to guess an amount of $2r$ bits. Although this method is no longer a polynomial time attack on RSA, it might be feasible in practice on todays computers for values up to approximately $r = 30$.

This result shows that it is dangerous to choose secret exponents d which are slightly above the Wiener bound. Similar extensions also hold for the Boneh-Durfee attack on small secret exponent RSA, which we describe in Section 7.2.

4.3 Generalizing Wiener's attack

"One should always generalize."
Carl Jacobi

The attack of Wiener applies to every public key (N, e) that satisfies the relation $ed - 1 = 0 \bmod \phi(N)$ for some $d \leq \frac{1}{3}N^{\frac{1}{4}}$. In this section, we will generalize Wiener's attack to all public keys (N, e), where e satisfies a relation $ew + z = 0 \bmod \phi(N)$ with small parameters w and z. In general, the bounds on w and z will depend on the size of e and the size of $p - q$. Wlog, we can assume in the following that $p - q \geq N^{\frac{1}{4}}$, since otherwise Fermat's factorization algorithm factors N in polynomial time.

In this section, we will prove the following main theorem. Note that the constants in this theorem are not optimized.

Theorem 20 *Given an RSA public key tuple (N, e), where $N = pq$. Suppose that e satisfies an equation $ew + z = 0 \bmod \phi(N)$ with*

$$0 < w \leq \frac{1}{3}\sqrt{\frac{\phi(N)}{e}}\frac{N^{\frac{3}{4}}}{p-q} \quad \text{and} \quad |z| \leq \frac{1}{8}\frac{e}{\phi(N)}\frac{p-q}{N^{\frac{1}{4}}} \cdot w.$$

Then N can be factored in time $\mathcal{O}(\log^2(N))$.

Let us consider this theorem in the light of standard RSA, where we have $p - q \geq cN^{\frac{1}{2}}$ for some $c < 1$. Since p and q are of the same bit-size, we also have an upper bound $p - q \leq N^{\frac{1}{2}}$ (see Section 2.1).

Using the above inequalities for our bounds on w and z yields the following corollary.

Corollary 21 *Given an RSA public key tuple (N, e), where $N = pq$ with $p - q \geq cN^{\frac{1}{2}}$ for some $c < 1$. Suppose that e satisfies an equation $ew + z = 0 \bmod \phi(N)$ with*

$$0 < w \leq \frac{1}{3}\sqrt{\frac{\phi(N)}{e}}N^{\frac{1}{4}} \quad \text{and} \quad |z| \leq \frac{c}{8}\frac{e}{\phi(N)}N^{\frac{1}{4}} \cdot w.$$

Then N can be factored in time $\mathcal{O}(\log^2(N))$.

Finally, we consider the normal RSA-case where $e \in \mathbb{Z}^*_{\phi(N)}$ which implies $e < \phi(N)$. Furthermore, we observe that $\frac{N^{\frac{1}{4}}}{\phi(N)} \geq N^{-\frac{3}{4}}$.

Corollary 22 *Given an RSA public key tuple (N, e), where $N = pq$, $e \in \mathbb{Z}^*_{\phi(N)}$ and $p - q \geq cN^{\frac{1}{2}}$ for some $c < 1$. Suppose that e satisfies an equation $ew + z = 0 \bmod \phi(N)$ with*

$$0 < w \leq \frac{1}{3} N^{\frac{1}{4}} \quad \textit{and} \quad |z| \leq \frac{c}{8} N^{-\frac{3}{4}} ew.$$

Then N can be factored in time $\mathcal{O}(\log^2(N))$.

Wiener's theorem is a special case of Corollary 22, where $w = d$ and $z = -1$. Since we do not fix the parameter z to the constant -1, we can apply our method to a much larger class of public keys.

The rest of this section is dedicated to the proof of Theorem 20. We combine the lattice-based method that we introduced in Section 4.2 with an approach of de Weger [70], who generalized Wiener's attack to arbitrary prime differences $p - q \geq N^{\frac{1}{4}}$.

As opposed to the proof of Theorem 16, we do not obtain the term $p + q$ directly from the lattice-based approach, since we have not fixed z to the constant -1. Thus, z introduces an additive error to the term $p + q$. We show, that an approximation of $p + q$ leads to an approximation of $p - q$. Combining both approximations gives us an approximation of p. Then, we use a theorem of Coppersmith (Theorem 11) to find the factorization of N.

Theorem 11 (Coppersmith) *Let $N = pq$ with $p > q$. Suppose we know an approximation $\tilde{p}$ of p with*

$$|\, p - \tilde{p}\,| \leq N^{\frac{1}{4}}$$

Then we can find the factorization of N in time polynomial in $\log N$.

Now we are ready to state the proof of Theorem 20.

Proof. [Theorem 20] We know that e satisfies an equation

$$ew + z = k\phi(N) \quad \text{for some } k \in \mathbb{Z}, \tag{4.2}$$

where

$$0 < w \leq \frac{1}{3}\sqrt{\frac{\phi(N)}{e}} \frac{N^{\frac{3}{4}}}{p - q} \quad \text{and} \quad |z| \leq \frac{1}{8} \frac{e}{\phi(N)} \frac{p - q}{N^{\frac{1}{4}}} \cdot w. \tag{4.3}$$

We want to argue that we can assume wlog that

$$\gcd(w, k) = 1. \tag{4.4}$$

By equation (4.2), every integer that divides both w and k must also divide z. Thus, if we assume that $\gcd(w,k) > 1$, we can divide equation (4.2) by $\gcd(w,k)$, obtaining an equivalent relation with even smaller parameters $w' = \frac{w}{\gcd(w,k)}$, $z' = \frac{z}{\gcd(w,k)}$ and $k' = \frac{k}{\gcd(w,k)}$, where w' and k' are co-prime.

Using $\phi(N) = (N+1-p-q)$, we rewrite equation (4.2) as

$$ew + k(p+q-2\sqrt{N}) + z = k(N+1-2\sqrt{N}). \tag{4.5}$$

This gives us a bivariate polynomial

$$f_M(x,y) = ex + y$$

with the root $(x_0, y_0) = (w, k(p+q-2\sqrt{N}) + z)$ modulo $M = N+1-2\sqrt{N}$.

The proof will now proceed in the following steps.

Roadmap of the proof

- We apply Coppersmith's method using the formulation of Howgrave-Graham to transform the polynomial $f_M(x,y)$ into a polynomial $f(x,y)$ satisfying $f(x_0,y_0) = 0$ over $\mathbb{Z}$ (and not just modulo M).
- We use f to extract the unknown parameters (x_0, y_0) and k.
- From (x_0, y_0) and k, we compute an approximation of $p+q$.
- From the approximation of $p+q$, we compute an approximation of $p-q$.
- Combining both approximation, we obtain an approximation of p which gives us the factorization of N using Coppersmith's Theorem (Theorem 11).

We recall Howgrave-Graham's theorem (Theorem 14) for our special case. Here, we reuse the formulation of Howgrave-Graham's theorem from Section 4.2.

Theorem 17 (Howgrave-Graham) *Let $f(x,y)$ be a polynomial that is a sum of at most two monomial. Suppose*

(1) $f(x_0,y_0) = 0 \bmod N$, where $|x_0| \le X$ and $|y_0| \le Y$

(2) $\|f(xX, yY)\| < \frac{1}{\sqrt{2}}N$

Then $f(x_0,y_0) = 0$ holds over the integers.

In order to apply Theorem 17, we need upper bounds for the unknown parameters x_0 and y_0. Let $X := \frac{1}{3}\sqrt{\frac{\phi(N)}{e}\frac{N^{\frac{3}{4}}}{p-q}}$. Then $x_0 = w \le X$ by definition.

Note that equation (4.3) clearly implies $|z| < \frac{1}{2}ew$. Since $k = \frac{ew+z}{\phi(N)}$, we can conclude that

$$\frac{1}{2}\frac{ew}{\phi(N)} \leq k \leq 2\frac{ew}{\phi(N)} \leq 2\frac{e}{\phi(N)}X. \tag{4.6}$$

In addition, de Weger [70] observed that $0 < p + q - 2\sqrt{N} \leq \frac{(p-q)^2}{4N^{\frac{1}{2}}}$. This can easily be seen by writing

$$(p-q)^2 = (p+q)^2 - 4N = (p+q-2\sqrt{N})(p+q+2\sqrt{N}).$$

It follows that $p + q - 2\sqrt{N} > 0$ and

$$p+q-2\sqrt{N} = \frac{(p-q)^2}{p+q+2\sqrt{N}} \leq \frac{(p-q)^2}{4\sqrt{N}},$$

which proves de Weger's inequality.

Using de Weger's inequality together with (4.6), we can provide an upper bound for y_0:

$$y_0 = k(p+q-2\sqrt{N}) - 1 \leq \frac{1}{2}\frac{e(p-q)^2}{\phi(N)N^{\frac{1}{2}}}X.$$

We define $Y := \frac{1}{2}\frac{e(p-q)^2}{\phi(N)N^{\frac{1}{2}}}X$. Now, our polynomial $f_M(x,y)$ satisfies condition *(1)* of Howgrave-Graham's theorem. The auxiliary polynomial $f_0(x,y) = Mx$ also satisfies the first condition, since it is the zero polynomial modulo M. Hence the integer linear combinations

$$g(x,y) = c_0 f_0(x,y) + c_1 f_M(x,y)$$

satisfy condition *(1)*. The coefficient vectors of $g(xX, yY)$ form a lattice L in $\mathbb{Z}^2$, where L is spanned by the row vectors of the (2×2)-lattice basis

$$B = \begin{bmatrix} MX & \\ eX & Y \end{bmatrix}.$$

We have to search among these coefficient vectors of $g(xX, yY)$ for a vector v which satisfies the second condition in Howgrave-Graham's theorem, namely $\|v\| < \frac{1}{\sqrt{2}}M$. Since we can find a shortest vector in a two-dimensional lattice using Gauss reduction, it suffices to show that L contains a suitably short vector.

Lemma 23 *The lattice L contains a vector v with $\|v\| < \frac{1}{\sqrt{2}}M$.*

Proof: By Minkowski's theorem (Theorem 3), we know that L contains a vector v with $\|v\| \leq \sqrt{2\det(L)}$. Hence, whenever the condition

$$\sqrt{2\det(L)} < \frac{1}{\sqrt{2}}M$$

is satisfied, then $\|v\| < \frac{1}{\sqrt{2}}M$. The condition is equivalent to $\det(L) < \frac{1}{4}M^2$. Since $\det(L) = MXY$, we can plug in the values

$$X = \frac{1}{3}\sqrt{\frac{\phi(N)}{e}\frac{N^{\frac{3}{4}}}{p-q}}\ , \quad Y = \frac{1}{2}\frac{e(p-q)^2}{\phi(N)N^{\frac{1}{2}}}X \ \text{ and } \ M = (N+1-2\sqrt{N})$$

and check the inequality. Thus, our condition simplifies to

$$\frac{1}{18}N \leq \frac{1}{4}(N+1-2\sqrt{N}).$$

But this condition holds whenever $N \geq 4$. This concludes the proof. $\square$

By Lemma 23, L contains a vector $v = (c_0, c_1) \cdot B$ with $\|v\| < \frac{1}{\sqrt{2}}M$. We interpret v as the coefficient vector of a polynomial $f(xX, yY)$ and apply Theorem 17. Thus, we know that $f(x_0, y_0) = 0$ over the integer. The following lemma shows how to extract the root (x_0, y_0).

Lemma 24 *Let $v = (c_0, c_1) \cdot B$ be a shortest vector in L. Then*

$$(x_0, y_0) = (|c_1|, |c_0M + c_1e|) \quad \textit{and} \quad k = |c_0|.$$

Proof: Applying Howgrave-Graham's Theorem, we obtain a polynomial $f(x, y)$ satisfying

$$f(x_0, y_0) = c_0Mx_0 + c_1f_M(x_0, y_0) = 0 \text{ over } \mathbb{Z}.$$

Using equation (4.5), we conclude that $f_M(x_0, y_0) = kM$. Reordering terms leads to

$$\frac{k}{x_0} = -\frac{c_0}{c_1}.$$

By construction, we know that the parameters k and x_0 are positive. Our goal is to show that the fractions on both sides of the equation are in their lowest terms, since then $k = |c_0|$ and $x_0 = |c_1|$. But $\gcd(k, x_0) = \gcd(k, w) = 1$ as we assumed wlog, see (4.4). Additionally we have $\gcd(c_0, c_1) = 1$, since v is a shortest vector in L.

Therefore, the coefficients c_0, c_1 of a shortest vector in L (in terms of the basis B) give us the secret parameters x_0 and k. Using equation (4.5), we can compute

$$y_0 = kM - ex_0 = |c_0|M - |c_1|e.$$

Notice, that exactly one of the coefficients c_0, c_1 is negative. Therefore $|y_0| = |c_0M + c_1e|$. But we know that $y_0 > 0$ which proves the claim. $\square$

By Lemma 24, we obtain the secret parameters (x_0, y_0) and k. Since $y_0 = k(p+q-2\sqrt{N}) + z$, we can compute the term

$$s = \frac{y_0}{k} + 2\sqrt{N} = p + q + \frac{z}{k},$$

which is an approximation of $p+q$ up to some additive error $\frac{z}{k}$. We bound the error term by using equations (4.3) and (4.6):

$$\left|\frac{z}{k}\right| \leq \frac{1}{8}\frac{e}{\phi(N)}\frac{p-q}{N^{\frac{1}{4}}}w \cdot 2\frac{\phi(N)}{ew} = \frac{p-q}{4N^{\frac{1}{4}}}.$$

Define $\Delta := \frac{p-q}{4N^{\frac{1}{4}}}$. Notice that in the case $p - q < 2N^{\frac{1}{4}}$ we have $\Delta < \frac{1}{2}$. Therefore, the integer closest to s equals $p+q$ and we can factor N immediately. On the other hand, we know that $p - q \leq N^{\frac{1}{2}}$, which bounds the error term: $\Delta \leq \frac{1}{4}N^{\frac{1}{4}}$.

Our goal is to show that an approximation s of $p+q$ with additive error at most Δ leads to an approximation $\tilde{s}$ of $p-q$ with error at most $\frac{3}{2}N^{\frac{1}{4}}$. Then, it follows that $\frac{s+\tilde{s}}{2}$ is an approximation of p up to an additive error of

$$\begin{aligned}
\left|\frac{s+\tilde{s}}{2} - p\right| &\leq \frac{1}{2}\left|s - p - q + \tilde{s} - p + q\right| \\
&\leq \frac{1}{2}\left|s - (p+q)\right| + \frac{1}{2}\left|\tilde{s} - (p-q)\right| \\
&\leq \frac{1}{2}\Delta + \frac{3}{4}N^{\frac{1}{4}} < N^{\frac{1}{4}}.
\end{aligned}$$

By Coppersmith's theorem (Theorem 11), an approximation of p with error at most $N^{\frac{1}{4}}$ leads to the factorization of N. Hence our approximation $\frac{s+\tilde{s}}{2}$ yields the desired factorization.

In order to conclude the proof of Theorem 20, it remains to show that we can indeed transform our approximation s of $p+q$ into an approximation $\tilde{s}$ of $p-q$ with the desired error bound. One can easily relate the terms $p+q$ and $p-q$ using

$$p - q = \sqrt{(p-q)^2} = \sqrt{p^2 - 2N + q^2} = \sqrt{(p+q)^2 - 4N}.$$

Since we have an approximation s of $p+q$, we can compute an approximation $\tilde{s} := \sqrt{s^2 - 4N}$ of $p-q$. The following lemma shows that $\tilde{s}$ is indeed a well-defined approximation of $p-q$ within the error bound $\frac{3}{2}N^{\frac{1}{4}}$. The lemma mainly uses the Mean Value Theorem.

Lemma 25 *Let $N = pq$ with $p - q \geq 2N^{\frac{1}{4}}$. Given an approximation of $p+q$ with error at most $\frac{p-q}{4N^{\frac{1}{4}}}$, one can find an approximation of $p-q$ with error at most $\frac{3}{2}N^{\frac{1}{4}}$ in polynomial time.*

Proof: As before, let $s := p+q+r$ be the approximation of $p+q$ with error $|r| \leq \frac{p-q}{4N^{\frac{1}{4}}}$. Define $\tilde{s} := \sqrt{s^2 - 4N}$. In order for $\tilde{s}$ to be well-defined, we need to show that $s^2 - 4N \geq 0$. We know that

$$s^2 - 4N = (p+q)^2 + 2r(p+q) + r^2 - 4N = (p-q)^2 + 2r(p+q) + r^2. \qquad (4.7)$$

It suffices to show that $2|r|(p+q) \leq (p-q)^2$. Instead we prove the slightly better bound

$$2|r|(p+q) \leq \frac{3}{4}(p-q)^2, \qquad (4.8)$$

which will be useful in the further context of the proof. Since $2|r| \leq \frac{p-q}{2N^{\frac{1}{4}}}$, we only have to show that $p+q \leq \frac{3}{2}N^{\frac{1}{4}}(p-q)$, but

$$\frac{2}{3}\frac{p+q}{N^{\frac{1}{4}}} \leq 2N^{\frac{1}{4}} \leq p-q,$$

where the last inequality follows by our precondition on the prime difference.

Now we claim that $|\tilde{s} - (p-q)| \leq \frac{3}{2}N^{\frac{1}{4}}$. For every continuous function $g(x)$ on the interval $[x_0, x_1]$ the Mean Value Theorem holds:

$$\exists x \in [x_0, x_1] \; : \; g'(x) = \frac{g(x_1) - g(x_0)}{x_1 - x_0},$$

where $g'(x)$ is the first derivate of $g(x)$.

Let $g(x) := \sqrt{x}$ be the square root function and define $x_0 := \min\{\tilde{s}^2, (p-q)^2\}$ and $x_1 = \max\{\tilde{s}^2, (p-q)^2\}$. The function $g(x)$ is continuous on the interval $[x_0, x_1]$, since $x_0 > 0$. Observe that $|x_1 - x_0| = 2r(p+q) + r^2$ by equation (4.7). Applying the Mean Value Theorem gives us

$$|\tilde{s} - (p-q)| = \frac{1}{2}\left|x^{-\frac{1}{2}}\right| \cdot \left|2r(p+q) + r^2\right|.$$

We have to bound the two terms on the right hand side. Using inequalities (4.7) and (4.8) yields

$$x \geq x_0 \geq (p-q)^2 + 2r(p+q) \geq \frac{1}{4}(p-q)^2.$$

Therefore we have $\frac{1}{2}|x^{-\frac{1}{2}}| \leq \frac{1}{p-q}$. In order to bound the second term, we observe that

$$r^2 \leq \frac{p-q}{4N^{\frac{1}{4}}}r \leq \frac{1}{4}N^{\frac{1}{4}}r < \frac{1}{4}(p+q)r.$$

Therefore, we bound the second term by

$$|2r(p+q) + r^2| \leq \frac{9}{4}|r|(p+q) \leq \frac{9}{16}\frac{p-q}{N^{\frac{1}{4}}}(p+q) < \frac{3}{2}(p-q)N^{\frac{1}{4}}.$$

Putting the inequalities for both terms together, we obtain

$$|\tilde{s} - (p - q)| \leq \frac{3}{2} N^{\frac{1}{4}},$$

which proves the lemma. □

Let us finally summarize the whole algorithm which leads to the factorization of the modulus.

Algorithm Generalized Wiener-Attack

INPUT: (N, e), where $N = pq$

1. Construct the lattice L with basis B.
2. Find a shortest vector $v = (c_0, c_1) \cdot B \in L$ using Gauss reduction.
3. Compute $(x_0, y_0) = (|c_1|, |c_0 N + c_1 e|)$ and $k = |c_0|$.
4. Compute $s = \frac{y_0}{k} + 2\sqrt{N}$ and $\tilde{s} = \sqrt{s^2 - 4N}$.
5. Run Coppersmith's Algorithm (Theorem 11) on input $\frac{s+\tilde{s}}{2}$.

OUTPUT: p, q

Since every step in the Algorithm Generalized Wiener-Attack can be done in polynomial time, this concludes the proof of Theorem 20. ◇

4.4 An application – Cryptanalysis of the YKLM-scheme

"It is the peculiar beauty of this method, gentlemen, and one which endears it to really scientific mind, that under no circumstance can it be of the smallest possible utility."
Henry John Stephen Smith (1826-1883)

At ICISC 2001, Yen, Kim, Lim and Moon [73, 74] presented an RSA-type scheme (briefly called the YKLM-scheme) that was designed to counteract the so-called Bellcore-attack by Boneh, DeMillo and Lipton [11] on CRT-RSA. CRT-RSA is an RSA variant, where the signature generation is done using the Chinese Remainder Theorem (CRT).

Boneh, DeMillo and Lipton showed that whenever exactly one of the signature computations either modulo p or modulo q is faulty, the signature reveals the factorization of N in polynomial time.

The idea of Yen, Kim, Lim and Moon was what they called "fault infective computation": In their scheme, the computations modulo p and q are designed to depend on each other such that either both computations are correct or both computations are faulty. This should prevent the Bellcore-attack.

Unfortunately, Yen, Kim, Lim and Moon need a special RSA key generation process in order to make their scheme efficient. Their public key e satisfies a relation with some small parameters that will be described later. The efficiency of the YKLM-scheme relies on the fact that these parameters are indeed much smaller than the modulus N. It was raised as an open question by the authors whether one could use random public keys e as well in their scheme by maintaining the same performance.

We show that the public keys constructed in the YKLM-scheme satisfy the conditions of Corollary 22, i.e., for every public exponent e we have $ew + z = 0 \bmod \phi(N)$ with small w and z.

Let us first review the modified key generation algorithm in the YKLM-scheme.

RSA Key Generation in the YKLM-scheme

Modulus : Randomly choose two primes p and q of the same bit-size and compute the product $N = pq$.

Small parameters : Fix a bound B, where $B \ll N$. Randomly choose e_r and r in $\{1, \ldots, B\}$ such that $\gcd(e_r, \phi(N)) = 1$.

Compute $d_r = e_r^{-1} \bmod \phi(N)$.

Secret exponent : Compute $d = d_r + r$. If $\gcd(d, \phi(N)) \neq 1$, choose different parameters e_r and r.

Public exponent : Compute $e = d^{-1} \bmod \phi(N)$.

Public parameters: Publish the tuple (N, e).

The authors of the YKLM-scheme pointed out that instead of the public key tuple (N, e) one could even publish the parameters e_r and r as well. The next theorem shows that the parameters e_r and r immediately reveal the factorization of N.

Theorem 26 *Given (N, e, e_r, r), the factorization of N can be found in probabilistic polynomial time in the bit-length of N.*

Proof: Consider the public key equation $ed - 1 = 0 \bmod \phi(N)$. The secret key d has a decomposition into the unknown part d_r and the known parameter r

$$e(d_r + r) - 1 = 0 \bmod \phi(N).$$

Multiplication with e_r removes the unknown parameter d_r

$$e(1 + e_r r) - e_r = 0 \bmod \phi(N).$$

Since every parameter on the left hand side is known, we can compute a multiple $k\phi(N)$ of the Euler function

$$e(1 + e_r r) - e_r = k\phi(N) \quad \text{for some } k \in \mathbb{N}. \tag{4.9}$$

By a straight-forward generalization of Theorem 2, such a multiple $k\phi(N)$ yields the factorization of N in probabilistic polynomial time in the bit-length of N. □

Certainly, there is no need to publish the small parameters e_r and r in the YKLM-scheme. On the other hand, the Wiener attack does not apply to the public key (N, e), since the construction of d ensures that d is large: In order to be an inverse of $e_r \le B$, d_r must be of size at least $\frac{\phi(N)}{B}$ and thus $d = d_r + r > \frac{\phi(N)}{B}$.

However, d has a special structure. It is close to some d_r which has a small inverse e_r modulo $\phi(N)$. We see by equation (4.9) that e thus satisfies a congruence

$$ew + z = 0 \bmod \phi(N),$$

with the small parameters $w = 1 + e_r r$ and $z = -e_r$. Therefore, we can apply our Generalized Wiener attack of Section 4.3. The following corollary follows from Corollary 22.

Corollary 27 *Let (N, e) be a public key tuple constructed by the key generation process in the YKLM-scheme with $p - q \ge cN^{\frac{1}{2}}$ for some $c < 1$. Furthermore, let e_r and r satisfy the conditions*

$$1 + e_r r \le \frac{1}{3} N^{\frac{1}{4}} \quad \textit{and} \quad e_r \le \frac{1}{16} c N^{\frac{1}{4}}.$$

Then N can be factored in time polynomial in $\log(N)$.

Proof: In order to be able to apply Corollary 22, it remains to show that $\frac{1}{16} c N^{\frac{1}{4}} \le \frac{1}{8} c N^{-\frac{3}{4}} e(1 + e_r r)$. Using equation (4.9), we conclude that

$$\frac{1}{8} c N^{-\frac{3}{4}} e(1 + e_r r) > \frac{1}{8} c N^{-\frac{3}{4}} \phi(N) > \frac{1}{16} c N^{\frac{1}{4}},$$

which proves the claim. □

Since the efficiency of the YKLM-scheme relies on the fact that e_r and r are very small compared to N, Corollary 27 breaks the YKLM-scheme for all reasonable parameter choices.

This result is — besides the Wiener and Boneh-Durfee attack — another warning to crypto-designers in order to be careful when using secret keys of a special structure in the RSA key generation process, especially if only small parameters are used to generate the keys. The secret key d must not be small itself in order to yield the factorization. If the corresponding public key e satisfies a modular relation $ew + z = 0 \bmod \phi(N)$ with small unknowns then this suffices to find the factors p and q.

4.5 There are $N^{\frac{3}{4}-\epsilon}$ weak keys

"Seek simplicity, and distrust it."
Alfred North Whitehead (1861-1947)

Let us first recall our definition of weak keys (Definition 15).

Definition 15 *Let C be a class of RSA public keys (N, e). The size of the class C is defined by*

$$size_C(N) := |\{e \in \mathbb{Z}^*_{\phi(N)} \mid (N, e) \in C\}|.$$

C is called weak *if:*

1. *$size_C(N) = \Omega(N^\gamma)$ for some $\gamma > 0$.*
2. *There exists a probabilistic algorithm A that on every input $(N, e) \in C$ outputs the factorization of N in time polynomial in* $\log(N)$.

The elements of a weak class are called weak keys.

In Section 4.3, we showed that every public key tuple (N, e) that satisfies a relation $ew + z = 0 \bmod \phi(N)$, with

$$0 < w \leq \frac{1}{3}\sqrt{\frac{\phi(N)}{e}}\frac{N^{\frac{3}{4}}}{p-q} \quad \text{and} \quad |z| \leq \frac{1}{8}\frac{e}{\phi(N)}\frac{p-q}{N^{\frac{1}{4}}} \cdot w \tag{4.10}$$

yields the factorization of N in polynomial time. Therefore the Generalized Wiener Attack from Section 4.3 defines a weak class C — notice that $\text{size}_C(N)$ must be polynomial in N since C contains as a subset the $N^{\frac{1}{4}-\epsilon}$ weak keys from Wiener's attack. The main question we will study here is, how large our weak class C is.

Question: What size has the weak class C of the Generalized Wiener Attack?

What bounds can we expect for $\text{size}_C(N)$? As a first estimate, we can sum over all tuples (w, z) within the bounds given by the inequalities in (4.10). This gives us an upper bound on the size of C. Therefore, we have at most

$$\text{size}_C(N) \leq \left(\frac{1}{3} \sqrt{\frac{\phi(N)}{e}} \frac{N^{\frac{3}{4}}}{p-q} \right)^2 \cdot \frac{1}{8} \frac{e}{\phi(N)} \frac{p-q}{N^{\frac{1}{4}}} = \frac{1}{72} \frac{N^{\frac{5}{4}}}{p-q} = \mathcal{O}\left(\frac{N^{\frac{5}{4}}}{p-q} \right) \tag{4.11}$$

weak keys for every fixed N. This is an upper bound on $\text{size}_C(N)$, since:

- Different tuples (w, z) might define the same public exponent e.
- Some of the tuples (w, z) do not even define a legitimate public exponent e, i.e., a key $e \in \mathbb{Z}^*_{\phi(N)}$.

Instead of an upper bound on $\text{size}_C(N)$, we are more interested in a lower bound. Namely, we want to know the minimal number of public exponents $e \in \mathbb{Z}^*_{\phi(N)}$ that yield the factorization for some fixed modulus N. This section is dedicated to the proof of a lower bound for $\text{size}_C(N)$.

As a result we obtain that our lower bound almost perfectly matches the upper bound. If $p - q = \Omega(N^{\frac{1}{4}+\gamma})$, $\gamma > 0$, we obtain a lower bound of

$$\text{size}_C(N) = \Omega\left(\frac{N^{\frac{5}{4}-\epsilon}}{p-q} \right),$$

where $\epsilon > 0$ is arbitrarily small for N suitably large.

Let us have a closer look at this result. In the standard RSA case, we have $p - q = \Omega(N^{\frac{1}{2}})$ which implies a bound of

$$\text{size}_C(N) = \Omega\left(N^{\frac{3}{4}-\epsilon} \right)$$

weak RSA key tuples (N, e) for every fixed N.

On the other hand, we know that Fermat's factorization algorithm yields the factorization of N in polynomial time if $p - q = \mathcal{O}(N^{\frac{1}{4}})$. But the number of weak keys for $p - q = N^{\frac{1}{4}+\gamma}$, $0 < \gamma \leq \frac{1}{4}$ is $\Omega(N^{1-\gamma-\epsilon})$. This means that the number of weak keys scales almost perfectly with the prime difference $p - q$. We have an interpolation property towards Fermat factorization: As $p - q$ decreases, more and more key tuples are weak and as γ approaches zero almost all keys are weak. This corresponds to the fact that for $\gamma = 0$, all tuples (N, e) are weak because one can find the factorization of N in polynomial time without any further information that is encoded in e.

We will now prove the lower bound result, where we use the following main lemma.

Lemma 28 *Let $f(N,e)$, $g(N,e)$ be functions on RSA public keys (N,e) such that*

$$f^2(N,e)g(N,e) < \phi(N), \ f(N,e) \geq 2 \text{ and } g(N,e) \leq f(N,e).$$

The number of public keys $e \in \mathbb{Z}^*_{\phi(N)}$, $e \geq \frac{\phi(N)}{4}$ that satisfy an equation $ew + z = 0 \bmod \phi(N)$ for $w \leq f(N,e)$ and $|z| \leq g(N,e)w$ is at least

$$\frac{f^2(N,e)g(N,e)}{8\log\log^2(N^2)} - \mathcal{O}(f^2(N,e)N^\epsilon),$$

where $\epsilon > 0$ is arbitrarily small for N suitably large.

Using Lemma 28, we can immediately prove our lower bound theorem.

Theorem 29 *Let $p-q = N^{\frac{1}{4}+\gamma}$ with $0 < \gamma \leq \frac{1}{4}$. Let C be the weak class that is given by the public key tuples (N,e), where e satisfies a relation of the form $ew+z = 0 \bmod \phi(N)$ such that w, z satisfy the bounds of (4.10). In addition, we restrict the public exponents to $e \in \mathbb{Z}^*_{\phi(N)}$ with $e \geq \frac{\phi(N)}{4}$. Then*

$$size_C(N) = \Omega\left(\frac{N^{1-\gamma}}{\log\log^2(N^2)}\right).$$

Proof: Using the bounds of (4.10), we define $f(N,e) := \frac{1}{3}\sqrt{\frac{\phi(N)}{e}\frac{N^{\frac{3}{4}}}{p-q}}$ and $g(N,e) := \frac{1}{8}\frac{e}{\phi(N)}\frac{p-q}{N^{\frac{1}{4}}}$. In order to satisfy the requirements of Lemma 28, we have to show that $f^2(N,e)g(N,e) < \phi(N)$, $f(N,e) \geq 2$ and $g(N,e) \leq f(N,e)$.

Using (4.11), we see that

$$f^2(N,e)g(N,e) = \frac{1}{72}\frac{N^{\frac{5}{4}}}{p-q} < \frac{1}{72}N < \phi(N).$$

Since $e < \phi(N)$ and $p - q \leq N^{\frac{1}{2}}$, we observe that $f(N,e) = \Omega(N^{\frac{1}{4}})$. Thus, $f(N,e) \geq 2$ for sufficiently large N.

Finally, $g(N,e) \leq f(N,e)$ imposes the condition

$$\frac{3}{8}\left(\frac{e}{\phi(N)}\right)^{\frac{3}{2}}\frac{(p-q)^2}{N} \leq 1,$$

which is satisfied since $e \leq \phi(N)$ and $(p-q)^2 \leq N$.

Now we can apply Lemma 28. Since $g(N,e) = \Omega(N^\gamma)$, the term $\frac{f^2(N,e)g(N,e)}{8\log\log^2(N^2)}$ dominates the error term $\mathcal{O}(f^2(N,e)N^\epsilon)$. Using $f^2(N,e)g(N,e) = \frac{1}{72}\frac{N^{\frac{5}{4}}}{p-q}$ and $p - q = N^{\frac{1}{4}+\gamma}$ proves the claim. $\square$

We obtain the following corollary immediately.

Corollary 30 *Let C be the weak class that is given by the public key tuples (N, e), where the public exponent $e \in \mathbb{Z}^*_{\phi(N)}, e \geq \frac{\phi(N)}{4}$ satisfies a relation of the form $ew + z = 0 \bmod \phi(N)$ such that w, z satisfy the bounds of (4.10). Then*

$$size_C(N) = \Omega\left(\frac{N^{\frac{3}{4}}}{\log\log^2(N^2)}\right).$$

It remains to prove Lemma 28. Let us first recall the preconditions: Throughout the proof, $f(N, e)$ and $g(N, e)$ will be functions such that $f^2(N, e)g(N, e) < \phi(N)$, $f(N, e) \geq 2$ and $g(N, e) < f(N, e)$. We will use the shorthand notations f and g for $f(N, e)$ and $g(N, e)$, respectively. We consider public keys that satisfy a relation $ew + z = k\phi(N)$ for some $k \in \mathbb{Z}$, where the parameters w and z satisfy the bounds $w \leq f$ and $|z| \leq gw$.

First, we note that looking for such $e \geq \frac{\phi(N)}{4}$ that satisfy an equation of the form $ew + z = k\phi(N)$ for arbitrary k is equivalent to looking for arbitrary e that satisfy an equation of the form $ew + z = k\phi(N)$ for reasonably large k.

Lemma 31 *Assume $e \in \mathbb{N}$ satisfies $ew - z = k\phi(N)$ for $w \leq f$, $|z| \leq gw$, and $k > \lceil \frac{f}{4} \rceil$, then $e \geq \frac{\phi(N)}{4}$.*

Proof: Using our preconditions on the functions f and g, we observe that

$$|z| \leq gw \leq gf < \frac{\phi(N)}{f} < \phi(N).$$

Hence $ew = k\phi(N) + z \geq (k-1)\phi(N)$. Now the bounds $k \geq \frac{f}{4} + 1$ and $w \leq f$ imply $e \geq \frac{f}{4w}\phi(N) \geq \frac{\phi(N)}{4}$. ◻

Roadmap of the proof

We define a set T which is a union of pairwise disjoint subsets $T(k)$. Each subset $T(k)$ consists of tuples (w, z) — w, z within the given bounds — such that

- Every $(w, z) \in T(k)$ defines a public key $e \in \mathbb{Z}^*_{\phi(N)}$ with $ew + z = k\phi(N)$.
- Different $(w, z) \in T$ (not only in $T(k)$!) define different public keys e.
- $|T| \geq \frac{f^2 g}{8 \log\log^2(N^2)} - \mathcal{O}(f^2 N^\epsilon)$

Let us define the sets $T(k)$ and their union T.

Definition 32 *For every fixed N and for every $k \in \mathbb{Z}$ define the set*

$$T(k) := \left\{ (w, z) \in \mathbb{N} \times \mathbb{Z} \;\middle|\; \begin{array}{c} k < w < f\,,\ |z| \leq gw, \\ z = k\phi(N) \bmod w, \gcd(w, k\phi(N)) = 1, \gcd(z, \phi(N)) = 1 \end{array} \right\}.$$

Further define
$$T := \cup_{k=\frac{f}{4}+1}^{f} T(k).$$

The following lemma shows that each tuple $(w, z) \in T(k)$ defines a public exponent $e \in \mathbb{Z}^*_{\phi(N)}$ of the desired form.

Lemma 33 *For every tuple $(w, z) \in T(k)$ there is an $e \in \mathbb{Z}_\phi(N)^*$ such that $ew + z = k\phi(N)$.*

Proof: Since $z = k\phi(N) \bmod w$ by Definition 32, there exists an integer e such that $ew + z = k\phi(N)$.

Using $0 < k < w$ and $|z| \leq gw < gf^2 < \phi(N)$, we obtain

$$0 < e = \frac{k\phi(N) - z}{w} \leq \frac{k\phi(N) - z}{k+1} < \phi(N).$$

Finally, since $\gcd(z, \phi(N))$, we have

$$\gcd(ew, \phi(N)) = \gcd(k\phi(N) - z, \phi(N)) = 1,$$

and therefore $\gcd(e, \phi(N)) = 1$, which concludes the proof. $\square$

Next we want to show that different tuples (w, z) lead to different public keys.

Lemma 34 *Let (w_0, z_0) and (w_1, z_1) be two different tuples from T. Then they define distinct public exponents.*

Proof: According to Lemma 33, there exist $e_0, e_1 \in \mathbb{Z}^*_{\phi(N)}$ such that $e_0w_0 + z_0 = 0 \bmod \phi(N)$ and $e_1w_1 + z_1 = 0 \bmod \phi(N)$. Assume for contradiction that $e_0 = e_1$. This implies
$$\frac{z_0}{w_0} = \frac{z_1}{w_1} \bmod \phi(N).$$
Note that the fractions are well-defined since $\gcd(w_0, \phi(N)) = 1$ and $\gcd(w_1, \phi(N)) = 1$ by Definition 32.

Equivalently we can write

$$z_0w_1 = z_1w_0 \bmod \phi(N). \tag{4.12}$$

But $w_0, w_1 \leq f$ and $z_0, z_1 \leq g$. This implies that both products in the identity above are smaller than $fg < \frac{\phi(N)}{f} < \frac{\phi(N)}{2}$ in absolute value. Thus, the identity even holds over the integers and not just modulo $\phi(N)$.

By Definition 32, all the tuples $(w, z) \in T(k)$ satisfy the relations $z = k\phi(N) \bmod w$ and $\gcd(w, \phi(N)) = 1$. This implies

$$\gcd(w, z) = \gcd(w, k\phi(N)) = 1.$$

Thus, we have $\gcd(w_0, z_0) = 1$ as well as $\gcd(w_1, z_1) = 1$. We conclude that $w_0 = w_1$ and $z_0 = z_1$ in equation (4.12), which is a contradiction. Therefore $e_0 \neq e_1$ and the claim follows. $\square$

It remains to derive a lower bound for $|T|$. Our goal is to provide a lower bound for the size of the sets $T(k)$. If the sets $T(k)$ are pairwise distinct then $|T|$ can be bounded by $\sum_{k=\frac{f}{4}+1}^{f} |T(k)|$. The pairwise distinctness is proven in the following lemma.

Lemma 35 *If $k \neq l$ then $T(k) \cap T(l) = \emptyset$.*

Proof: Assume for contradiction that $(w, z) \in T(k), T(l)$ for $k \neq l$. Then

$$z = k\phi(N) \bmod w \quad \text{and} \quad z = l\phi(N) \bmod w.$$

Since $\gcd(w, \phi(N)) = 1$, we can divide both equations by $\phi(N)$. We conclude that $k = l \bmod w$. But $k, l < w$ by Definition 32 and therefore $k = l$ over the integers, contradicting the assumption $k \neq l$. $\square$

In order to provide a lower bound for $|T(k)|$, we fix the parameters N, k and w and sum over all z in $T(k)$. The following technical lemma from the area of analytic number theory gives us a bound for the number of integers z that meet the requirements of Definition 32. Since we want to apply the lemma for different variable settings, we give it in a general form and introduce new parameters l, r and u. To understand the connection to the definition of $T(k)$, the reader can substitute the variables l, r by the left and right bound $-gw$ and gw, respectively. The variable u can be substituted by $k\phi(N)$ in the following lemma.

Lemma 36 *Let N, $w \in \mathbb{N}$ with $\gcd(w, \phi(N)) = 1$. Let $l, r, u \in \mathbb{Z}$ be arbitrary. The set*

$$\{z \in \mathbb{Z} \mid l < z \leq r, z = u \bmod w \text{ and } \gcd(z, \phi(N)) = 1\}$$

contains at least

$$\frac{1}{2\log\log(\phi(N))} \frac{r-l}{w} - \mathcal{O}(N^{\epsilon})$$

elements, where $\epsilon > 0$ is arbitrarily small for suitably large N.

Proof: To prove Lemma 36 we need the Moebius μ-function. For $m = p_1 p_2 \cdots p_k \in \mathbb{N}$, $p_1, \dots, p_k$ prime, it is defined as

1. $\mu(1) = 1$
2. $\mu(m) = (-1)^k$ if all the primes $p_1, \dots, p_k$ are different.
3. $\mu(m) = 0$ otherwise

Let $z_0 \in \mathbb{Z}_w$ be the unique integer that satisfies $z_0 = u \bmod w$.

Then z, $l < z \leq r$ satisfies $z = u \bmod w$ iff z is of the form $z_0 + tw$ with $\frac{l-z_0}{w} < t \leq \frac{r-z_0}{w}$. A divisor d of $\phi(N)$ divides $z_0 + tw$ iff $t = -\frac{z_0}{w} \bmod d$. Note that the inverse of w modulo d exists since $w \in \mathbb{Z}^*_{\phi(N)}$. In the interval $(\frac{l-z_0}{w}, \frac{r-z_0}{w}]$ there are at least $\lfloor \frac{r-l}{dw} \rfloor$ numbers of this form. By the inclusion-exclusion principle we conclude that the number we are looking for is at least

$$\begin{aligned}\sum_{d|\phi(N)} \mu(d) \lfloor \frac{r-l}{dw} \rfloor &= \sum_{d|\phi(N)} \mu(d) \frac{r-l}{dw} - \sum_{d|\phi(N)} \mu(d) \frac{r-l \bmod dw}{dw} \\ &\geq \frac{r-l}{w} \sum_{d|\phi(N)} \frac{\mu(d)}{d} - \sum_{d|\phi(N)} |\mu(d)|\end{aligned}$$

Since $\displaystyle\sum_{d|\phi(N)} \frac{\mu(d)}{d} = \frac{\phi(\phi(N))}{\phi(N)}$ and $\displaystyle\frac{\phi(\phi(N))}{\phi(N)} \geq \frac{1}{2\log\log(\phi(N))}$ for N large enough, we get

$$\sum_{d|\phi(N)} \mu(d) \lfloor \frac{r-l}{dw} \rfloor \geq \frac{1}{2\log\log(\phi(N))} \frac{r-l}{w} - \sum_{d|\phi(N)} |\mu(d)|.$$

Finally we use the fact that $\displaystyle\sum_{d|\phi(N)} |\mu(d)| = \mathcal{O}(\phi^{\epsilon}(N)) = \mathcal{O}(N^{\epsilon})$ for all $\epsilon > 0$ (see [1]). $\square$

If we set $w = 1$, we obtain

Corollary 37 *Let $N \in \mathbb{N}$ and $l, r \in \mathbb{Z}$. The number of integers z with $l < z \leq r$ and* $\gcd(z, \phi(N)) = 1$ *is at least*

$$\frac{r-l}{2\log\log(\phi(N))} - \mathcal{O}(N^{\epsilon}),$$

where $\epsilon > 0$ is arbitrarily small for suitably large N.

We are now able to prove a lower bound for $|T| = \sum_{k=\frac{f}{4}+1}^{f} |T(k)|$.

Lemma 38 *Let g, f be as defined in Lemma 28. Then*

$$|T| \geq \frac{f^2 g}{8 \log\log^2(N^2)} - \mathcal{O}(f^2 N^\epsilon),$$

where $\epsilon > 0$ is arbitrarily small for suitably large N.

Proof: From Lemma 36 we conclude that for fixed N, k and w the number of $z \in \mathbb{Z}, |z| \leq gw$ such that $(w,z) \in T(k)$ is at least

$$\frac{g}{\log\log(N)} - \mathcal{O}(N^\epsilon).$$

Hence

$$\begin{aligned}
\sum_{k=\frac{f}{4}+1}^{f} |T(k)| &\geq \sum_{k=\frac{f}{4}+1}^{f} \sum_{\substack{w=k+1 \\ \gcd(w,k\phi(N))=1}}^{f} \left(\frac{g}{\log\log(N)} - \mathcal{O}(N^\epsilon) \right) \\
&\geq \left(\frac{g}{\log\log(N)} \sum_{k=\frac{f}{4}+1}^{f} \sum_{\substack{w=k+1 \\ \gcd(w,k\phi(N))=1}}^{f} 1 \right) - \mathcal{O}(f^2(N) N^\epsilon).
\end{aligned}$$

Applying Corollary 37 shows that for $\epsilon > 0$ arbitrary small and N large enough

$$\sum_{\substack{w=k+1 \\ \gcd(w,k\phi(N))=1}}^{f} 1 \geq \frac{f-k}{2\log\log(k\phi(N))} - \mathcal{O}(N^\epsilon).$$

Hence

$$\begin{aligned}
\sum_{k=\frac{f}{4}+1}^{f} \sum_{\substack{w=k+1 \\ \gcd(w,k\phi(N))=1}}^{f} 1 &\geq \left(\sum_{k=\frac{f}{4}+1}^{f} \frac{f-k}{2\log\log(kN)} \right) - \mathcal{O}(fN^\epsilon) \\
&\geq \left(\frac{1}{2\log\log(N^2)} \sum_{k=0}^{\frac{3}{4}f-1} k \right) - \mathcal{O}(fN^\epsilon) \\
&\geq \frac{f^2}{8\log\log(N^2)} - \mathcal{O}(fN^\epsilon).
\end{aligned}$$

Since $g \leq f$, the lemma follows. $\square$

5 Unbalanced RSA with Small CRT-Exponent

"Problems worthy of attack,
prove their worth by fighting back."
Anonymous

5.1 Introduction

In Chapter 4, we studied classes of weak keys (N, e) in RSA. Weak keys give an attacker enough information to factor N in polynomial time. The practically most important class of weak keys is the one introduced by Wiener [71] (see Section 4.2): If the secret key d corresponding to (N, e) satisfies $d < \frac{1}{3}N^{\frac{1}{4}}$ then (N, e) is a weak key. This class was extended by Boneh-Durfee [12] (see Section 7.2) to all d with $d < N^{0.292}$ using Coppersmith's heuristic method for solving modular bivariate polynomial equations.

The importance of these classes comes from the fact that the decryption/signature process in RSA requires the computation of $m^d \bmod N$. The cost of computing this term is $\mathcal{O}(\log d \log^2 N)$. Hence, one might by tempted to use small exponents d in order to speed up the decryption/signature process. Assume a situation where a device with poor computational power (e.g. a smart-card) has to compute signatures frequently, whereas the validation of the signatures is done by a high-speed computer. It is preferable to perform the largest amount of the computation on the fast device by choosing a large public exponent e with corresponding small secret exponent d. Unfortunately as shown by Boneh-Durfee, this leads to a completely insecure scheme whenever $d < N^{0.292}$.

These results show that one cannot use a small decryption exponent d in RSA in order to speed up the decryption process. On the other hand, Wiener's and Boneh-Durfee's attack do not affect the security of the fast Quisquater-Couvreur decryption variant (see Section 2.1): One can use a decryption exponent d such that $d_p := d \bmod p-1$ and $d_q := d \bmod \frac{q-1}{2}$ are both small[1]. Such an exponent d is called a *small CRT-exponent* in the following. In order to sign a message m, one computes $m^{d_p} \bmod p$ and $m^{d_q} \bmod q$. Both terms are combined using the Chinese Remainder Theorem to yield the desired

[1]Here we use the term $\frac{q-1}{2}$ instead of $q-1$, since we will later need in our key generation process that the moduli $p-1$ and $\frac{q-1}{2}$ are coprime

term $m^d \bmod N$. The attacks of Wiener and Boneh-Durfee do not work in this case, since d is likely to be large.

It is an open problem if there is a polynomial time algorithm that breaks RSA if d_p and d_q are small. This problem is mentioned several times in the literature, see e.g. [9, 12, 71]. The best algorithm that is known runs in time $\mathcal{O}(\min(\sqrt{d_p}, \sqrt{d_q}))$ which is exponential in the bit-size.

Goal: Find polynomial time attacks for (N, e) with corresponding small CRT-exponent.

In this chapter, we give the first polynomial time attacks on RSA with small CRT-exponent. Unfortunately, our results are restricted to the case of unbalanced prime numbers p and q. The use of unbalanced primes was first proposed by Shamir [60] to guard the modulus N against different kinds of factorization algorithms and to speed up the computations in RSA.

There are also other systems that use unbalanced primes. In 2000, Modadugu, Boneh and Kim [52] proposed an easy RSA key generation protocol on an untrusted server by using unbalanced primes. At Asiacrypt 1999, Sun, Yang and Laih [66] introduced an RSA-variant with unbalanced primes p and q that was designed to counteract the Wiener and Boneh-Durfee attack. Interestingly, it turns out that the use of unbalanced primes in their scheme even decreases the security. At Asiacrypt 2000, Durfee and Nguyen [28] showed that a variant of the Boneh-Durfee attack works for larger exponents d than the original attack if the prime factors are unbalanced. The Durfee-Nguyen approach breaks the Sun-Yang-Laih scheme in two out of three of the suggested parameter settings.

We show in the following chapter that there is also a decrease in security for unbalanced primes when using small CRT-exponents. The more unbalanced the prime factors are, the larger are the CRT-exponents that can be attacked by our methods.

Let $q < N^{\beta}$. We show in Section 5.2 that an RSA public key tuple (N, e) with corresponding d_p satisfying the condition

$$d_p \leq \frac{1}{2} N^{\frac{1-3\beta}{2}}$$

yields the factorization of N in time $\mathcal{O}(\log^2(N))$. Since for $\beta > \frac{1}{3}$ the exponent of N becomes negative, this method only works provided that $q < N^{\frac{1}{3}}$. In terms of the notation of Chapter 4, the new attack gives us a class of weak keys of size $\Omega(N^{\frac{1-3\beta}{2}-\epsilon})$.

Our approach is based on Coppersmith's technique [20] in the modular multivariate case (see Section 3.4). More precisely, we use a modular bivariate polynomial equation with a small root. This root gives us the factorization of N. Using Howgrave-Graham's theorem (Theorem 14), we turn the modular bivariate polynomial into a polynomial $f(x, y)$ over $\mathbb{Z}$ such that the desired small root must be among the roots of $f(x, y)$. Interestingly, for our polynomial $f(x, y)$ we are able to prove that this small root can be extracted easily. This shows that our method provably factors the modulus N. Note that this is in contrast to many other approaches using the multivariate method [3, 12, 28, 37] which rely on the resultant heuristic.

The attack in Section 5.2 uses a two-dimensional lattice. In Section 5.3, we generalize our method to lattices of arbitrary dimension. This improves the condition above to

$$d_p \leq N^{\frac{1-3\beta+\beta^2}{2}-\epsilon}$$

for some arbitrary small error term $\epsilon > 0$. Therefore, this approach works as long as $\beta < \frac{3-\sqrt{5}}{2} = \hat{\phi}^2 \approx 0.382$, where $\hat{\phi} = \frac{1-\sqrt{5}}{2}$ is the conjugate of the golden ratio. Again, we can show that the desired root can be extracted in polynomial time. This yields a rigorous method for factoring N.

In Section 5.4, we use a different modular bivariate polynomial. Unfortunately, this time we cannot give a rigorous proof for the method. It relies on Coppersmith's resultant heuristic for modular multivariate polynomials.

This approach works for larger CRT-exponents than our first attack. It is applicable whenever

$$d_p \leq N^{1-\frac{2}{3}(\beta+\sqrt{\beta^2+3\beta})-\epsilon}$$

For the parameter choice $\beta \geq \frac{3}{8}$, the exponent of N becomes negative. Hence, the method works only for $q < N^{\frac{3}{8}}$, which is slightly worse than the previous bound $N^{0.382}$. But note that for $\beta \to 0$ the method works for almost all d_p. This means that the number of weak keys (N, e) approaches N when the prime factors are completely unbalanced. This seems to be a natural result: One expects that factoring N becomes easier when the prime factors are more unbalanced. However, this intuition does not hold for all known factorization algorithms. For instance, the running time of the asymptotically best known factorization algorithm — the Number Field Sieve — does not depend on the size of the smallest prime factor but on the size of N.

Interestingly, our approaches only make use of the fact that the prime factors are unbalanced and that d_p is suitably small. They do not require that the value $d_q = d \bmod \frac{q-1}{2}$ is chosen to be small, which is a reasonable choice in order to speed up the Quisquater-Couvreur method. Contradicting to our intuition, small values of d_q do not yield better bounds in our attacks. This raises the question whether it is possible to derive attacks with improved bounds that can profit from small values of d_q as well. Does this lead to attacks on RSA with small CRT-exponent even in the case of balanced primes? At the moment, this is an open question.

Key generation using the Chinese Remainder Theorem (CRT)

We briefly describe the key generation process when using the Quisquater-Couvreur method with unbalanced prime factors. In our scenario, the RSA modulus N is composed of a large prime factor p and a small prime factor q. The secret decryption exponent d is chosen to be small modulo $p-1$ and of arbitrary size modulo $q-1$.

CRT Key Generation Process

Fix a bit-size n for the public key modulus N. Additionally, fix two positive parameters β, δ with $\beta \leq \frac{1}{2}$ and $\delta \leq 1$.

Modulus: Randomly choose prime numbers p and q with bit-sizes approximately $(1-\beta)n$ and βn, respectively. Additionally, $p-1$ and $\frac{q-1}{2}$ must be coprime. Compute the modulus $N = pq$. If the smaller prime factor q does not satisfy $q < N^{\beta}$, repeat the prime generation.

Secret exponent: Choose a small secret $d_p \in \mathbb{Z}^*_{p-1}$ such that $d_p \leq N^{\delta}$. Choose another secret $d_q \in \mathbb{Z}^*_{\frac{q-1}{2}}$ arbitrarily.

Chinese remaindering: Compute the unique $d \bmod \frac{\phi(N)}{2}$ that satisfies $d = d_p \bmod p-1$ and $d = d_q \bmod \frac{q-1}{2}$.

Public exponent: Compute the inverse e of d in $\mathbb{Z}^*_{\frac{\phi(N)}{2}}$.

Public parameters: Publish the tuple (N, e).

In this chapter, we study the following problem.

Question: Up to which parameter choices for β and δ does the public key tuple (N, e) yield the factorization of N?

Note that the decryption and the signature generation process of a message m are very efficient for small β and δ. Since d_p is small, the computation of $m^{d_p} \bmod p-1$ requires only a small number of multiplications. Furthermore, the computation of $m^{d_q} \bmod \frac{q-1}{2}$ is cheap because q is small. Both terms can easily be combined to yield the desired term $m^d \bmod \frac{\phi(N)}{2}$ using the Chinese Remainder Theorem (CRT).

In the next section, we will show that given the public key (N, e) there is a provable polynomial time algorithm that factors N if the condition $d_p \leq \frac{1}{2}N^{\frac{1-3\beta}{2}}$ holds. This implies that our method works as long as $\beta < \frac{1}{3}$. The smaller β is chosen, the larger values of d_p can be attacked. For $\beta = 0$, we obtain $d_p < \frac{1}{2}\sqrt{N}$. Later, we will improve the bound for β up to $\frac{3-\sqrt{5}}{2} \approx 0.382$ (see Section 5.3) and for d_p up to almost all d_p when β is suitably small (see Section 5.4).

5.2 An approach for $q < N^{\frac{1}{3}}$

In this section, we prove the following theorem.

Theorem 39 *Given an RSA public key tuple (N, e) with $N = pq$ and secret exponent d. Let $q < N^\beta$ and*

$$d_p \le \frac{1}{2} N^{\frac{1-3\beta}{2}}.$$

Then N can be factored in time $\mathcal{O}(\log^2(N))$.

Before we prove Theorem 39, let us illustrate our result in Figure 5.1. Let $\delta := \log_N(d_p)$ denote the size of d_p in terms of the size of N. The area under the graph is the feasible region for our attack. We see that $\delta \to \frac{1}{2}$ as the prime factors get more and more imbalanced. On the other hand, our attack does not work for $\beta \ge \frac{1}{3}$.

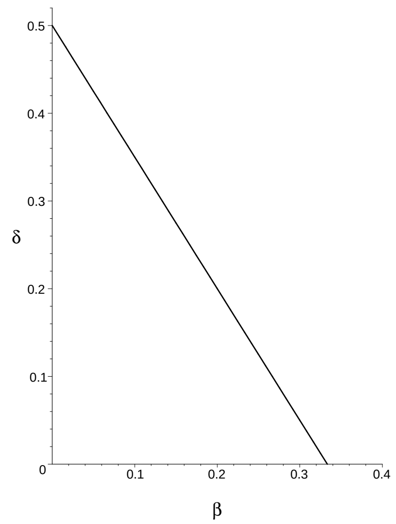

Figure 5.1: The approach for $q < N^{\frac{1}{3}}$

Proof. Assume we are given a public key (N, e) that is constructed according to the CRT Key Generation process of Section 5.1. We know that

$$ed_p = 1 \bmod p - 1.$$

Thus, there is an integer $k \in \mathbb{N}$ such that

$$ed_p - 1 = k(p-1) \quad \text{over } \mathbb{Z}. \tag{5.1}$$

We can rewrite this equation as

$$ed_p + (k-1) = kp \tag{5.2}$$

In the following, we assume that q does not divide k. Otherwise, the right hand side of the equation is a multiple of N and we can obtain much stronger results. This case will be analyzed later (see Theorem 43).

Equation (5.2) gives us the polynomial

$$f_p(x,y) = ex + y$$

with a root $(x_0, y_0) = (d_p, k-1)$ modulo p.

Let $X := \frac{1}{2}N^{\frac{1-3\beta}{2}}$. By construction, we have $d_p \leq X$. Since $e < \frac{(p-1)(q-1)}{2}$, we further obtain

$$|k-1| < |k| = \left|\frac{ed_p - 1}{p-1}\right| < \frac{ed_p}{p-1} < \frac{q-1}{2}d_p < N^{\beta}X. \tag{5.3}$$

Let $Y := N^{\beta}X$ denote this upper bound. Then, we have a modular bivariate polynomial equation f_p with a small root (x_0, y_0) that satisfies $|x_0| \leq X$ and $|y_0| \leq Y$. This modular equation can be turned into an equation over the integers using Howgrave-Graham's theorem (Theorem 14). We recall the theorem for the special bivariate case that we need in the following.

Theorem 40 (Howgrave-Graham) *Let $f(x,y)$ be a polynomial that is a sum of at most two monomial. Suppose*

(1) $f(x_0, y_0) = 0 \bmod p$, where $|x_0| \leq X$ and $|y_0| \leq Y$

(2) $\|f(xX, yY)\| < \frac{1}{\sqrt{2}}p$

Then $f(x_0, y_0) = 0$ holds over the integers.

Using our polynomial $f_p(x,y)$, we want to construct a polynomial $f(x,y)$ that satisfies both conditions of Howgrave-Graham's theorem. We use the auxiliary polynomial $f_0(x) = Nx$ that also has the root x_0 modulo p, since N is a multiple of p. Therefore, every integer linear combination of f_0 and f_p has the root (x_0, y_0) modulo p. We construct a lattice L_p that is spanned by the coefficient vectors of the polynomials $f_0(xX)$ and $f_p(xX, yY)$. These coefficient vectors are the row vectors of the following (2×2)-lattice basis B_p.

$$B_p = \begin{bmatrix} NX & \\ eX & Y \end{bmatrix}$$

The following lemma shows that the lattice L_p always contains a vector v with norm smaller than $\frac{p}{\sqrt{2}}$. Since L is a two-dimensional lattice, we can find v using the Gauss reduction algorithm (see [58]). The vector v can then be transformed into a polynomial $f(x,y)$ satisfying $f(x_0, y_0) = 0$ over $\mathbb{Z}$.

Lemma 41 *L_p contains a smallest vector v with $\|v\| < \frac{p}{\sqrt{2}}$.*

Proof: By Minkowski's theorem (Theorem 3), L_p must contain a vector v with $\|v\| \leq \sqrt{2\det(L_p)}$. Thus, v has norm smaller than $\frac{p}{\sqrt{2}}$ if the condition

$$\sqrt{2\det(L_p)} < \frac{p}{\sqrt{2}}$$

holds.
Utilizing $\det(L_p) = NXY$ gives the condition $NXY < \frac{p^2}{4}$. Plugging in the values $X = \frac{1}{2}N^{\frac{1-3\beta}{2}}$ and $Y = N^{\beta}X$, this condition can be further transformed into

$$NXY = N^{1+\beta}X^2 = \frac{1}{4}N^{2-2\beta} < \frac{p^2}{4}.$$

Since $q < N^{\beta}$, we know that $p > N^{1-\beta}$. Thus the condition is satisfied and the claim follows. $\square$

Assume we have found a vector v in L_p with norm smaller than $\frac{p}{\sqrt{2}}$ by Gauss reduction. Let v be the coefficient vector of the polynomial $f(xX, yY)$. Applying Theorem 40, we know that $f(x, y)$ has a root $(x_0, y_0) = (d_p, k-1)$ over the integers. The next theorem shows that the root (x_0, y_0) can easily be determined when v is represented in terms of the basis B_p.

Lemma 42 *Let $v := (c_0, c_1) \cdot B_p$ be a shortest vector in L_p with $\|v\| < \frac{p}{\sqrt{2}}$. Then $|c_0| = k$ and $|c_1| = qd_p$.*

Proof: We have $v = c_0(NX, 0) + c_1(eX, Y)$. Define the polynomial $f(xX, yY)$ that has the coefficient vector v. By construction, $\|f(xX, yY)\| < \frac{p}{\sqrt{2}}$ and we can apply Theorem 40.

Therefore, the polynomial

$$f(x, y) = c_0Nx + c_1(ex + y)$$

has the root (x_0, y_0) over $\mathbb{Z}$. Plugging (x_0, y_0) into the equation yields

$$c_0Nx_0 = -c_1(ex_0 + y_0).$$

We know that $(x_0, y_0) = (d_p, k - 1)$. That leads to

$$c_0Nd_p = -c_1(ed_p + (k - 1)).$$

Using equation (5.2) and dividing by p gives us

$$c_0 q d_p = -c_1 k.$$

Since we assumed that q does not divide k, we have $\gcd(qd_p, k) = \gcd(d_p, k)$. Now, let us look at equation (5.1). Every integer that divides both d_p and k must also divide 1. Hence, $\gcd(d_p, k) = 1$.

Thus, we obtain

$$c_0 = ak \quad \text{and} \quad c_1 = -aqd_p$$

for some integer a. But v is a shortest vector in L_p. Therefore, we must have $|a| = 1$ and the claim follows. □

In the previous analysis, we made the assumption that q does not divide k. If we are in the very unlikely case that $k = qr$ for some $r \in \mathbb{Z}$, then we obtain analogous to the reasoning before the following stronger result.

Theorem 43 *Given an RSA public key tuple (N, e) with $N = pq$ and secret exponent d. Let $q < N^{\beta}$,*

$$k = qr \quad \text{and} \quad d_p \leq \frac{1}{4} N^{\frac{1-\beta}{2}}.$$

Then N can be factored in time $\mathcal{O}(\log^2(N))$.

Proof: We only sketch the proof, since the reasoning is completely analogous to the proof for the case where q does not divide k.

The polynomial $f_p(x, y) = ex + y$ has the root $(x_0, y_0) = (d_p, k - 1)$ not just modulo p but also modulo N. Thus, we can use the modulus N in Theorem 40. Analogous to Lemma 41, we conclude that L_p has a shortest vector v with norm smaller than $\frac{N}{\sqrt{2}}$ provided that $d_p \leq \frac{1}{4} N^{\frac{1-\beta}{2}}$.

Following the proof of Lemma 42, we see that $v = (c_0, c_1) \cdot B_p$ with $|c_0| = r$ and $|c_1| = d_p$. Since $\frac{ed_p - 1}{r} = q(p-1)$ by equation (5.1), the computation $\gcd(\frac{ed_p-1}{r}, N) = q$ reveals the factorization. □

Interestingly, choosing $\beta = \frac{1}{2}$ in Theorem 43 gives us the bound $d_p \leq \frac{1}{4} N^{\frac{1}{4}}$, which is similar to Wiener's bound in the attack on low secret exponent RSA (see Section 4.2).

Let us briefly summarize the whole factorization algorithm of Theorem 39.

Algorithm (Mod p)-Attack for unbalanced RSA with small CRT-exponent d_p

INPUT: (N, e), where $N = pq$ with $p > N^{1-\beta}$ and $d_p \leq \frac{1}{4}N^{\frac{1-3\beta}{2}}$

1. Construct the lattice basis B_p of L_p.
2. Find a shortest vector $v = (c_0, c_1) \cdot B_p$ in L_p using Gauss reduction.
3. If $\gcd(N, |c_1|) > 1$, compute $q = \gcd(N, |c_1|)$ and $p = \frac{N}{q}$. (Case: q does not divide k)
4. Else compute $q = \gcd(N, \frac{e|c_1|-1}{|c_0|})$ and $p = \frac{N}{q}$. (Case: q divides k)

OUTPUT: p, q

The total running time for Gauss reduction and greatest common divisor computations is $\mathcal{O}(\log^2(N))$. This completes the proof of Theorem 39. $\diamond$

5.3 Improving the bound to $q < N^{0.382}$

Using Theorem 39, our approach with the two-dimensional lattice L_p only works provided that $q < N^{\frac{1}{3}}$. In this section, we use lattices of larger dimensions to make our method work for less unbalanced moduli. We are able to improve the bound up to

$$q < N^{\frac{3-\sqrt{5}}{2}} \approx N^{0.382}.$$

In Section 5.2, we used Howgrave-Graham's theorem (Theorem 14) for bivariate polynomials for the special case $m = 1$, which gave us a two-dimensional lattice. Here, we generalize our approach by allowing arbitrary m.

The rest of the section is devoted to the proof of the following theorem.

Theorem 44 *For every fixed $\epsilon > 0$ there exists an integer N_0 such that for every $N > N_0$ the following holds:*
Given an RSA public key tuple (N, e) with $N = pq$ and secret exponent d. Let $q < N^{\beta}$ and

$$d_p \leq N^{\frac{1-3\beta+\beta^2}{2}-\epsilon}.$$

Then we can find the factorization of N in time polynomial in $\log(N)$.

Before we prove Theorem 44, let us illustrate our improved result in Figure 5.2. As before, we denote by $\delta := \log_N(d_p)$ the size of d_p in terms of the size of N. The area under the graph is the feasible region for our attack. For comparison we also draw in Figure 5.2 the line from our result in the previous section.

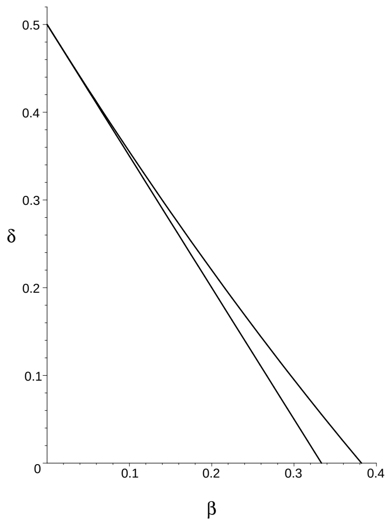

Figure 5.2: Improving the bound to $q < N^{0.382}$

Proof. Let $f_p(x, y) := ex + y$ be defined as in Section 5.2, i.e., f_p has the root $(x_0, y_0) = (d_p, k-1)$ modulo p. Furthermore, let $\delta := \frac{1-3\beta+\beta^2}{2} - \epsilon$. From the proof of Theorem 39, we know that $|x_0| \leq N^\delta$ and $|y_0| \leq N^{\beta+\delta}$.

Define the x-shifted polynomials

$$g_{m,i,j}(x, y) := N^{\max(0,m-j)} x^i f_p^j(x, y).$$

Note, that every integer linear combination of polynomials $g_{m,i,j}$ has the root (x_0, y_0) modulo p^m.

Let us define the bounds $X := \frac{n+1}{2} N^\delta$ and $Y := \frac{n+1}{2} N^{\beta+\delta}$. Furthermore, we fix a lattice dimension n. Next, we build a lattice $L_p(n)$ of dimension n using as basis vectors the coefficient vectors of $g_{m,i,j}(xX, yY)$ for $j = 0 \ldots n-1$ and $i = n-j-1$. The parameter m is a function of n and must be optimized.

For example, take $n = 4$ and $m = 2$. The lattice $L_p(n)$ is spanned by the row vectors of the following (4×4)-matrix

$$B_p(4) = \begin{bmatrix} N^2X^3 & & & \\ eNX^3 & NX^2Y & & \\ e^2X^3 & 2eX^2Y & XY^2 & \\ e^3X^3 & 3e^2X^2Y & 3eXY^2 & Y^3 \end{bmatrix}.$$

Note, that the lattice L_p of Section 5.2 is equal to $L_p(2)$.

In order to apply Howgrave-Graham's theorem (Theorem 14), we need a coefficient vector v with norm smaller than $\frac{p^m}{\sqrt{n}}$. The following Lemma shows that we can always find such a vector in $L_p(n)$ using L^3-reduction.

Lemma 45 *On input $B_p(n)$, the L^3-reduction outputs a vector $v \in L_p(n)$ with norm smaller than $\frac{p^m}{\sqrt{n}}$, where m is a function of n.*

Proof: An easy computation shows that

$$\det(L_p(n)) = N^{\frac{m(m+1)}{2}}(XY)^{\frac{n(n-1)}{2}} = \left(\frac{n+1}{2}\right)^{n(n-1)} N^{\frac{m(m+1)}{2}+(2\delta+\beta)\frac{n(n-1)}{2}}$$

for $m < n$. By the L^3-theorem (Theorem 4), the L^3-algorithm will find a vector v in $L_p(n)$ with

$$\|v\| \leq 2^{\frac{n-1}{4}} \det(L_p(n))^{\frac{1}{n}}.$$

Using $p > N^{1-\beta}$, we must satisfy the condition

$$2^{\frac{n-1}{4}} \det(L_p(n))^{\frac{1}{n}} \leq \frac{N^{(1-\beta)m}}{\sqrt{n}}.$$

We plug in the value for $\det(L_p(n))$ and obtain the inequality

$$N^{\frac{m(m+1)}{2}+(2\delta+\beta)\frac{n(n-1)}{2}} \leq cN^{(1-\beta)mn},$$

where the factor $c = \left(\left(2^{-\frac{3}{4}}(n+1)\right)^{n-1}\sqrt{n}\right)^{-n}$ does not depend on N. Thus, c contributes to the error term ϵ and will be neglected in the following.

We obtain the condition

$$\frac{m(m+1)}{2} + (2\delta+\beta)\frac{n(n-1)}{2} - (1-\beta)mn \leq 0.$$

Using straightforward arithmetic to minimize the left hand side, one obtains that $m = (1-\beta)n$ is asymptotically optimal for $n \to \infty$. Again doing some calculations, we finally end up with the condition $\delta \leq \frac{1-3\beta+\beta^2}{2} - \epsilon$, which is satisfied by construction. This concludes the proof. □

Now, we can use the above Lemma 45 in combination with Howgrave-Graham's theorem (Theorem 14) to construct a bivariate polynomial $g(x, y)$ with at most n monomials and root (x_0, y_0). The problem is how to extract the root (x_0, y_0).

Analogous to Lemma 42, one can show for a vector $v = (c_1, c_2, \ldots, c_n) \cdot B_p(n)$ with norm smaller than $\frac{p^m}{\sqrt{n}}$ that k divides c_1 and d_p divides c_n. But we may not be able to find these factors k and d_p easily.

Therefore, we use another method to obtain the root. This method is described in the following lemma.

Lemma 46 *Let $X := \frac{n+1}{2} N^{\delta}$ and $Y := \frac{n+1}{2} N^{\beta+\delta}$. Let $f_p(x, y) := ex+y$ be a polynomial with root (x_0, y_0) modulo p that satisfies $|x_0| \leq N^{\delta}$, $|y_0| \leq N^{\beta+\delta}$. Let v be a vector in $L_p(n)$ with norm smaller than $\frac{p^m}{\sqrt{n}}$, where v is the coefficient vector of a polynomial $g(xX, yY)$. Then, the polynomial*

$$h(x, y) = y_0 x - x_0 y \in \mathbb{Z}[x, y]$$

must divide $g(x, y)$. We can find $h(x, y)$ by factoring $g(x, y)$ over $\mathbb{Q}[x, y]$.

Proof: The point (x_0, y_0) is a root of f_p. For every integer a, the point (ax_0, ay_0) is also a root of f_p. Every root (ax_0, ay_0) with $|a| \leq \frac{n+1}{2}$ satisfies the conditions $|ax_0| \leq X$ and $|ay_0| \leq Y$ of Howgrave-Graham's Theorem. These are at least $n + 1$ roots. According to Howgrave-Graham's theorem, g must contain these roots over $\mathbb{Z}$.

But these roots lie on the line $y = \frac{y_0}{x_0} x$ through the origin. Hence, they are also roots of the polynomial

$$h(x, y) = y_0 x - x_0 y \in \mathbb{Z}[x, y].$$

Note, that h is an irreducible polynomial of degree 1 and f is a polynomial of degree n. Using the Theorem of Bézout (see [59], page 20), either g and h share at most n points or h must divide g. But we know $n+1$ common points of g and h. Thus, the polynomial h must divide g.

Since h is irreducible, we can find an integer multiple $h' = (by_0)x - (bx_0)y$ of h by factoring g over $\mathbb{Q}[x, y]$. Note that $\gcd(x_0, y_0) = 1$ since by equation (5.2) we know that $\gcd(d_p, k - 1)$ must divide kp, but $\gcd(d_p, kp) = \gcd(d_p, k) = 1$. Hence, we obtain h by computing $h = \frac{h'}{\gcd(by_0, bx_0)}$. ◻

Summarizing the results in this section, we obtain the following factorization algorithm.

Improved (Mod p)-Attack for unbalanced RSA with small CRT-exponent d_p

INPUT: (N, e), where $N = pq$ with $p > N^{1-\beta}$ and $d_p \leq N^{\frac{1-3\beta+\beta^2}{2}-\epsilon}$

1. Fix an integer n (depending on $\frac{1}{\epsilon}$) and construct the lattice basis $B_p(n)$ of $L_p(n)$.
2. Apply the L^3-algorithm to $B_p(n)$. Let v be a shortest vector that is found be the algorithm.
3. Construct from v the corresponding polynomial $g(x, y)$. Factor $g(x, y)$ over $\mathbb{Q}[x, y]$.
4. For every irreducible factor $h(x, y)$ of the form $ax + by$: Compute $x_0 = \frac{b}{\gcd(a,b)}$ and $y_0 = \frac{a}{\gcd(a,b)}$ and test whether
   - $\gcd(N, ex_0 + y_0) = p$ (Case: q does not divide k) or
   - $\gcd(N, y_0 + 1) = q$ (Case: q divides k).

OUTPUT: p, q

It is known that the factorization of the polynomial $g(x, y) \in \mathbb{Q}[x, y]$ can be done in (deterministic) time polynomial in $\log(N)$ (see [32, 38]). Note that the coefficients of $g(x, y)$ must be of bit-size polynomial in $\log(p)$ since the coefficient vector of $g(xX, yY)$ has norm smaller than $\frac{p^m}{\sqrt{n}}$. This concludes the proof of Theorem 44. $\diamond$

In practice, the factorization of polynomials over $\mathbb{Q}[x, y]$ is very fast. Thus, our method is practical even for large n.

5.4 An approach that allows larger values of d_p

Throughout this section, we assume that e is of the same order of magnitude as N. The results in this section as well as the results in Sections 5.2 and 5.3 can be easily generalized to arbitrary exponents e. Analogous to the results of Wiener [71] and Boneh-Durfee for small secret exponent RSA, the smaller the exponent e is, the better our methods work. On the other hand, one can completely counteract the attacks by adding to e a suitably large multiple of $\phi(N)$.

Since we use in the following Coppersmith's resultant heuristic for modular bivariate polynomial equations, our approach is heuristic as well. However, the only heuristic part consists of the assumption that a certain resultant computation yields a non-zero polynomial. In order to state our result as a theorem, we assume that this resultant computation never fails.

Assumption 47 *The resultant computation in the method of Theorem 48 never yields a non-zero polynomial.*

We carried out several experiments that support our assumption: We could not find an example where the resultant was zero.

Theorem 48 *Under Assumption 47, for every fixed $\epsilon > 0$ there exists an integer N_0 such that for every $N > N_0$ the following holds:*
Given an RSA public key tuple (N, e) with $N = pq$ and secret exponent d. Let $q < N^\beta$ and

$$d_p \le N^{1-\frac{2}{3}(\beta+\sqrt{3\beta+\beta^2})-\epsilon}.$$

Then we can find the factorization of N in time polynomial in $\log(N)$.

Proof. Let us reuse the equation $ed_p - 1 = k(p-1)$ and rewrite it as

$$(k-1)(p-1) + p = ed_p.$$

Multiplying with q yields

$$(k-1)(N-q) + N = ed_p q$$

This gives us the polynomial

$$f_e(y, z) = y(N-z) + N$$

with a root $(y_0, z_0) = (k-1, q)$ modulo e.

Let $\delta := 1 - \frac{2}{3}(\beta + \sqrt{3\beta + \beta^2}) - \epsilon$. Then we can define the upper bounds $Y := N^{\beta+\delta}$ and $Z := N^\beta$. Note that $|y_0| \le Y$ (see inequality (5.3)) and $|z_0| \le Z$. Analogous to Section 5.2, we can define a three-dimensional lattice L_e that is spanned by the row vectors of the (3×3)-matrix

$$B_e := \begin{bmatrix} e & & \\ & eY & \\ N & NY & -YZ \end{bmatrix}.$$

Using a similar argumentation as in Section 5.2, one can find a vector $v \in L_e$ with norm smaller than the bound $\frac{e}{\sqrt{3}}$ of Howgrave-Graham's Theorem provided that $d_p \le N^{\frac{1-3\beta}{2}-\epsilon}$. Hence as before, this approach does not work if $\beta \ge \frac{1}{3}$ or $d_p \ge N^{\frac{1}{2}}$. In Section 5.3, we used x-shifted polynomials to improve the bound for β. Now, z-shifted polynomials will help us to improve the bound for d_p up to $d_p < N^{1-\epsilon}$ (for $\beta \to 0$), or for β up to $\beta < \frac{3}{8}$ (for $d_p \to 0$).

Fix an integer m. Let us define the y-shifted polynomials

$$g_{i,j}(y,z) := e^{m-i} y^j f_e^i(y,z)$$

and the z-shifted polynomials

$$h_{i,j}(y,z) := e^{m-i} z^j f_e^i(y,z).$$

All these polynomials have the common root (y_0, z_0) modulo e^m. Thus, every integer linear combination of these polynomials also has the root (y_0, z_0).

We construct a lattice $L_e(m)$ that is defined by the span of the coefficient vectors of the y-shifted polynomials $g_{i,j}(yY, zZ)$ and $h_{i,j}(yY, zZ)$ for certain parameters i, j. We take the coefficient vectors of $g_{i,j}$ for all non-negative i, j with $i + j \leq m$ and the coefficient vectors $h_{i,j}$ for $i = 0 \ldots m$ and $j = 1 \ldots t$ for some t. The parameter t has to be optimized as a function of m.

For example, choose $m = 2$ and $t = 1$. We take the coefficient vectors of $g_{0,0}$, $g_{0,1}$, $g_{1,0}$, $g_{0,2}$, $g_{1,1}$, $g_{2,0}$ and the coefficient vectors of $h_{0,1}$, $h_{1,1}$, $h_{2,1}$ to build the lattice basis $B_e(2)$:

$$\left[\begin{array}{ccccccccc}
e^2 & & & & & & & & \\
 & e^2Y & & & & & & & \\
-eN & eNY & -eYZ & & & & & & \\
 & & & e^2Y^2 & & & & & \\
 & -eNY & & eN^2Y^2 & -eY^2Z & & & & \\
N^2 & -2N^2Y & 2NYZ & N^2Y^2 & -2NY^2Z & Y^2Z^2 & & & \\
\hline
 & & & & & & e^2Z & & \\
 & & eNYZ & & & & -eNZ & -eYZ^2 & \\
 & & -2N^2YZ & & N^2Y^2Z & -2NY^2Z^2 & N^2Z & 2NYZ^2 & Y^2Z^3
\end{array}\right]$$

The row vectors of $B_e(2)$ span the lattice $L_e(2)$.

In order to apply Theorem 40, we need a vector in $L_e(m)$ with norm smaller than $\frac{e^m}{\sqrt{\dim L_e(m)}}$. The following lemma shows that we can find two sufficiently short vectors using L^3-reduction.

Lemma 49 *On input $B_e(m)$, the L^3-algorithm outputs two vectors $v_1, v_2 \in L_e(m)$ with norm smaller than $\frac{e^m}{\sqrt{\dim L_e(m)}}$.*

Proof: A straightforward computation shows that

$$\det L_e(m) = (eY)^{\frac{1}{6}(2m^3+(6+3t)m^2+(4+3t)m)} Z^{\frac{1}{6}(m^3+(3+3t)m^2+(2+6t+3t^2)m+3t+3t^2)}.$$

Let $t := \tau m$ and $e = N^{1-o(1)}$. Using $Y = N^{\beta+\delta}$ and $Z = N^{\beta}$, we obtain

$$\det L_e(m) = N^{\frac{1}{6}m^3((\beta+\delta+1)(2+3\tau)+\beta(1+3\tau+3\tau^2)+o(1))}.$$

Analogous to the reasoning in Lemma 45, we apply the L^3-theorem (Theorem 4) and obtain the condition

$$\det L_e(m) < cN^{(1-o(1))m \dim L_e(m)},$$

where c does not depend on N and contributes to the error term ϵ. An easy calculation shows that $\dim(L) = \frac{(m+1)(m+2)}{2} + t(m+1)$. We plug in the values for $\det L_e(m)$ and $\dim L_e(m)$. Neglecting all low order terms yields the condition

$$3\beta(\tau^2 + 2\tau + 1) + \delta(3\tau + 2) - 3\tau - 1 < 0$$

for $m \to \infty$. Using elementary calculus to minimize the left hand side, we obtain an optimal choice for the value $\tau = \frac{1-2\beta-\delta}{2\beta}$. Plugging in this value, we finally end up with the condition

$$\delta \leq 1 - \frac{2}{3}(\beta + \sqrt{3\beta + \beta^2}),$$

which is satisfied by our construction. This concludes the proof of Lemma 49. $\square$

Using Lemma 49, we can apply Howgrave-Graham's theorem (Theorem 14) and obtain two polynomials $f_1(y, z)$, $f_2(y, z)$ with the common root (y_0, z_0) over $\mathbb{Z}$. But in contrast to the previous sections, we are not able to give a rigorous method to extract this root. Instead, we follow the resultant heuristic due to Coppersmith.

We take the resultant of f_1 and f_2 with respect to y. The resultant $r(z)$ is a polynomial in z that has the root z_0. By Assumption 47, $r(z)$ is not the zero polynomial. Thus, we obtain the unknown $z_0 = q$ by applying standard root finding algorithms to $r(z)$.

We summarize our results in the following factorization algorithm.

(Mod e)-Attack for unbalanced RSA with small CRT-exponent d_p

INPUT: (N, e), where $N = pq$ with $p > N^{1-\beta}$ and $d_p \leq N^{1-\frac{2}{3}(\beta+\sqrt{3\beta+\beta^2})-\epsilon}$

1. Fix an integer m and construct the lattice basis $B_p(m)$ of $L_p(m)$.
2. Apply the L^3-algorithm to $B_p(m)$. Let v_1, v_2 be the shortest vectors that are found by the algorithm.
3. Construct from v_1, v_2 the corresponding polynomials $f_1(y, z)$, $f_2(y, z)$ and compute $r(z) = res_y(f_1, f_2)$. If $r(z) = 0$ output "Failed".
4. For every root z_0 of $r(z)$: Test whether $z_0 = q$ and $\frac{N}{z_0} = p$.

OUTPUT: p, q

The running time of the algorithm is polynomial in $\log(N)$, which concludes the proof of Theorem 48. $\diamond$

We do not know if our lattice based approach yields the optimal bound. But there is a heuristic argument that gives us an upper bound for our method when using the polynomial $f_e(y, z)$.

Assume that the function $h(y, z) = y(N - z) \bmod e$ takes on random values in $\mathbb{Z}_e$ for $|y| \leq Y$ and $|z| \leq Z$. Every tuple (y, z) with $h(y, z) = -N \bmod e$ is a root of f_e. The expected number of those tuples is $\Omega(\frac{YZ}{e}) = \Omega(N^{2\beta+\delta-1})$. As soon as $2\beta+\delta-1$ is larger than some positive fixed constant, the number of small roots satisfying f_e is exponential in $\log(N)$. All these roots fulfill the criteria of Howgrave-Graham's Theorem. But we require that our polynomials only have poly$(\log(N))$ many roots in order to extract the desired root in polynomial time.

Thus heuristically, we cannot expect to obtain a bound better than $d_p \leq N^{1-2\beta}$ using the polynomial f_e. This implies that this approach cannot be used for the case of balanced RSA, where $\beta = \frac{1}{2}$.

It is an open problem if one can really reach this bound.

5.5 Comparison of the methods

We compare the methods introduced in Section 5.3 and Section 5.4. Let $\delta := \log_N(d_p)$ denote the size of d_p in terms of the size of N. In the figure below, we plotted the maximal δ as a function of β for which our two approaches succeed. The first method (Section 5.3) is represented by the line $\delta = \frac{1-3\beta+\beta^2}{2}$ resulting from Theorem 44. The second approach (Section 5.4) gives us the curve $\delta = 1 - \frac{2}{3}(\beta + \sqrt{3\beta + \beta^2})$ by Theorem 48. The areas below the curves represent the feasible region of parameter choices for our attacks. We see that our second method yields much better results for small β.

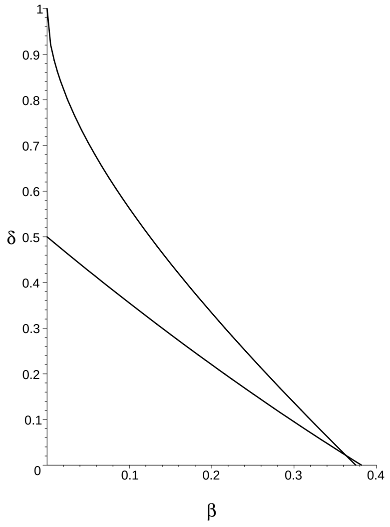

Figure 5.3: Comparison of the methods

One might be tempted to combine the two approaches and use the polynomials $ez \cdot f_p(x, y)$ and $N \cdot f_e(y, z)$ in a single lattice basis (i.e., working modulo eN). However, such a lattice will always contain an extremely short coefficient vector corresponding to the polynomial $f(x, y, z) = exz + y(N - z) - z$ over $\mathbb{Z}$. But this polynomial can be obtained by multiplying equation (5.1) with q and does not help us any further. It is an open problem if there is a successful way to combine the methods.

6 Knowing A Fraction of the Secret Key Bits

"The joy of suddenly learning a former secret and the joy of suddenly discovering a hitherto unknown truth are the same to me — both have the flash of enlightenment, the almost incredibly enhanced vision, and the ecstasy and euphoria of released tension."
Paul R. Halmos

6.1 Introduction

In the previous chapters, we studied weak keys (N, e) in RSA. All of our weak keys had the common property that e satisfied a relation with small unknown parameters modulo $\phi(N)$ (Chapter 4) or modulo $p-1$ (Chapter 5). This enabled us to compute either $p+q$ or p from (N, e) in polynomial time, which in turn lead to the factorization of N. But what can an adversary do if (N, e) does not happen to be a weak key? How can he gain information about the factorization of N?

We know that the secret key d satisfies the relation $ed = 1 \bmod \phi(N)$. Thus, if an attacker succeeds to get bits of d, he also obtains useful information about $\phi(N)$. Assume that he gets d completely, then he can factor N immediately using the randomized algorithm of Theorem 2. But the interesting question is whether he really needs to know all of the bits of d or just a (small) fraction of the secret key bits.

Question: When does a fraction of d provide enough information to factor N?

This question was introduced by Boneh, Durfee and Frankel [14] in 1999. In addition to its important theoretical interest, this is a very natural question arising from the intensive research efforts in cryptanalysis considering side-channel attacks on RSA (e.g. timing attacks, fault attacks, power analysis (see for instance [41, 42])).

Motivation: Side-Channel Attacks
In many adversary scenarios, an attacker using a side-channel attack either succeeds to obtain the most significant bits (MSBs) or the least significant bits (LSBs) of d in

consecutive order. Whether he gets MSBs or LSBs depends on the different ways of computing an exponentiation with d during the decryption process.

Let $d_n d_{n-1} \dots d_0$ be the binary representation of d. We briefly sketch two different variants of the Repeated Squaring Method, which was introduced in Section 2.1. Afterwards we give an example how a side-channel attack might work in order to recover bits of d.

Algorithm UP

INPUT: $m, N, d = d_n d_{n-1} \dots d_0$

1. Set $z := 1$.
2. For $i = 0$ to n
   a) If $(d_i = 1)$ set $z := z \cdot m \bmod N$
   b) Set $m := m^2 \bmod N$.

OUTPUT: $z = m^d \bmod N$

Algorithm DOWN

INPUT: $m, N, d = d_n d_{n-1} \dots d_0$

1. Set $z := 1$.
2. For $i = n$ to 0
   a) Set $z := z^2 \bmod N$.
   b) If $(d_i = 1)$ set $z := z \cdot m \bmod N$

OUTPUT: $z = m^d \bmod N$

Algorithm UP has the invariant that in the i^{th} execution of its loop $(i = 0, \dots, n)$, the intermediate result $m^{\sum_{j=0}^{i} 2^j d_j}$ is computed. Similar, we have in Algorithm DOWN the loop invariant that in the i^{th} execution $(i = 0, \dots, n)$, we obtain the intermediate result $m^{\sum_{j=n-i}^{n} 2^{j-(n-i)} d_j}$.

Our goal is to show that in a reasonable fault model an attacker can successively obtain the MSBs of d if Algorithm UP is used. This is a result due to Boneh, deMillo and Lipton [11]. Blömer and Otto [8] showed that an attacker can also get the LSBs in consecutive order if the computation is done according to Algorithm DOWN. However here, we want to focus just on the first case.

Assume that an attacker can ask a decryption device to sign an arbitrary message m. Furthermore, he is able to induce a single bit fault in the register that holds the variable z during the i^{th} execution of the loop. He has no control of the position of the faulty bit, but he has an exact control of the loop iteration i by timing his attack.

We want to show that in this case, the attacker can successively determine the MSBs of d. Assume that he has already determined the bits $d_n \dots d_{i+1}$ and he wants to get the bit d_i. Therefore, he induces a bit fault in the $(i-1)^{th}$ loop execution. We know by the loop invariant that up to this step, the register for z holds the intermediate result $m^{\sum_{j=0}^{i-1} 2^j d_j}$. Now, we get an additive fault of $\pm 2^b$, where b is the unknown bit position of the fault. The remaining computation is a multiplication with $m^{\sum_{j=i}^{n} d_j 2^j}$. Hence, the faulty signature S' has the form

$$S' = \left(m^{\sum_{j=0}^{i-1} 2^j d_j} \pm 2^b \right) \cdot m^{\sum_{j=i}^{n} d_j 2^j} \bmod N.$$

Therefore, S' differs from the correct signature m^d by the term

$$S' - m^d = \pm 2^b m^{\sum_{j=i}^{n} d_j 2^j} \bmod N.$$

Notice that we can guess the term on the right-hand side, since there are at most $\lceil \log N \rceil$ possibilities for the choice of b and we already know the bits $d_n \dots d_{i+1}$. Therefore, we just have to guess one more bit d_i. One can test whether the guess is correct by checking that

$$\left(S' \pm 2^b m^{\sum_{j=i}^{n} d_j 2^j} \right)^e = m \mod N.$$

The attack described here is just an example of a possible side-channel attack on RSA. There are many others attacks that determine consecutive bits of d. Therefore we think that it is reasonable to focus on the case where an adversary gets either MSBs or LSBs of d, and we ignore attacks where an adversary has to recover both sorts of bits or intermediate bits.

Some side-channel attacks are able to reveal a fraction of the secret key bits, but may fail to reveal the entire key (see [27]). For instance, the adversary might be restricted to a limited number of queries to the decryption device. In another reasonable scenario, an attacker might get bits only with a certain probability, but — similar to our example above — the probability that d_i is correct depends on the probability that the attacker's hypothesis of $d_n \dots d_{i+1}$ is correct. Hence, it gets harder and harder for him to recover additional bits. Therefore, it is essential to know how many bits of d suffice to discover the whole secret information d or equivalently the factorization of N.

Beside the theory of these side-channel attacks, it is an important theoretical question which security level an n-bit secret exponent really provides. It is also an interesting relaxation of the factorization problem: Without knowing bits, we assume that factoring is hard, which means that we do not expect the existence of a polynomial time algorithm for the problem. But with a sufficiently large fraction of the bits of d the problem becomes efficiently solvable. Hence, one can view a known fraction of d as a hint how to factor.

The results of Boneh, Durfee and Frankel

Surprisingly, in the case of LSBs, Boneh, Durfee and Frankel [14] showed that for low public exponent RSA (e.g. $e = \text{poly}(\log N)$) only a quarter of the bits of d are sufficient to find the factorization of N in polynomial time. Their method makes use of Coppersmith's theorem for known LSBs of p (Theorem 12): Given half of LSBs of p, the factorization of N can be found in polynomial time.

Let us state the result of Boneh, Durfee and Frankel for known LSBs of d.

Theorem 50 (BDF: LSBs) *Let $N = pq$ be an n-bit RSA modulus with $N = 3 \bmod 4$ and let $e < \frac{1}{8}N^{\frac{1}{4}}$. Given the*

$$\frac{n}{4} \textit{ LSBs of } d,$$

then N can be factored in time polynomial in $\log(N)$ and e.

Considering known MSBs of d, Boneh, Durfee and Frankel present an algorithm that works for $e \leq N^{\frac{1}{2}}$, again using Theorem 12. Here, the performance of the attack substantially depends on the fact whether one knows the factorization of e or not.

Theorem 51 (BDF: MSBs) *Let $N = pq$ be an n-bit RSA modulus.*

1. *Suppose e is a u-bit prime in the range $[N^{\frac{1}{4}}, N^{\frac{1}{2}}]$. Given the*

$$u \textit{ MSBs of } d,$$

   then N can be factored in time polynomial in $\log N$.

2. *Suppose e is a u-bit product of r distinct primes with known factorization and e is in the range $[N^{\frac{1}{4}}, N^{\frac{1}{2}}]$. Given the*

$$u \textit{ MSBs of } d,$$

   then N can be factored in time polynomial in $\log N$ and 2^r.

3. *Suppose $e \leq N^{\frac{1}{2}}$ is a u-bit number of unknown factorization. Further, suppose that $d > \epsilon N$ for some $\epsilon > 0$. Given the*

$$n - u \textit{ MSBs of } d,$$

   then N can be factored in time polynomial in $\log N$ and $\frac{1}{\epsilon}$.

4. Suppose $e \leq N^{\frac{1}{2}}$ *is a u-bit number of unknown factorization. Further, suppose that* $d > \epsilon N$ *and* $|p-q| > \epsilon\sqrt{N}$ *for some* $\epsilon > 0$*. Given the*

$$\frac{3}{4}n \text{ MSBs of } d,$$

then N *can be factored in time polynomial in* $\log N$ *and* $\frac{1}{\epsilon}$*.*

The last result (Theorem 51, 4.) is not explicitly mentioned in the original work [14] of Boneh, Durfee and Frankel but can be easily derived from another work by the same authors (see [15]).

It was raised as a challenging open question in the paper of Boneh, Durfee and Frankel whether there are polynomial time algorithms that find the factorization of N for values of e substantially larger than $N^{\frac{1}{2}}$ given only a subset of the secret key bits. The answer of this question is our first goal in this chapter.

Goal 1: Suppose $e > N^{\frac{1}{2}}$, find the factorization of N using only a fraction of d.

In this chapter, we provide algorithms for both cases MSBs and LSBs when $e > N^{\frac{1}{2}}$. Here is a brief overview of the results that we obtain.

MSBs of d known:
We present a method that works for all public exponents e in the interval $[N^{\frac{1}{2}}, N^{0.725}]$. The number of bits of d that have to be known increases with e. Let us provide some numerical examples of the required bits: For $e = N^{0.5}$ one has to know half of the MSBs of d, for $e = N^{0.55}$ a 0.71-fraction suffices whereas for $e = N^{0.6}$ a fraction of 0.81 is needed to factor N.

In contrast to Boneh, Durfee and Frankel, we do not use Coppersmith's theorem for known bits of p. Instead we directly apply Coppersmith's method for finding roots of modular trivariate polynomial equations (see Section 3.4). Since the method relies on the resultant heuristic in the multivariate case, our result is a heuristic as well. However, we provide various experiments that confirm the heuristic's reliability: None of our experiments failed to yield the factorization of N.

In Figure 6.1 we illustrate our result for MSBs. The size of the fraction of the bits that is sufficient in our attack is plotted as a function of the size of the public exponent e. We express the size of e in terms of the size of N (i.e., we use $\log_N(e)$). For a comparison with previous results, we also include in our graphs the results of Boneh, Durfee and Frankel. The marked regions in Figure 6.1 are the feasible regions for the various approaches.

Note that the area belonging to Theorem 51 (1. and 2.) requires that the factorization of e is known.

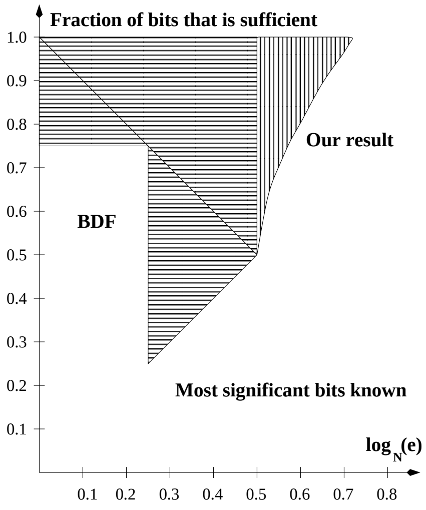

Figure 6.1: The results for known MSBs of d.

LSBs of d known:

We start by proving a result for all but a negligible fraction of the public exponents $e < N^{\frac{1}{2}}$. In contrast, the result of Boneh, Durfee and Frankel (Theorem 50) leads to a polynomial time algorithm only for exponents e of the order $\text{poly}(\log N)$. Our approach uses a 3-dimensional lattice to find the factorization of N using a single lattice basis reduction, whereas the method of Theorem 50 requires about e lattice reductions. We tested our attack with the frequently used RSA exponent $e = 2^{16} + 1$. Our algorithm is very fast but requires more bits of d than the method in Theorem 50.

Interestingly, our approach makes use of the linear independence of two sufficiently short vectors in the lattice and we do not need to apply Coppersmith's heuristic in this case. This makes our method rigorous and at the same time introduces a new method how to solve modular multivariate polynomial equations of a special form, thereby preventing the resultant heuristic.

In the next step, we generalize the 3-dimensional approach to multi-dimensional lattices. This improves the bound up to all $e < N^{\frac{7}{8}}$, which is the largest known bound for e in partial key exposure attacks on RSA. Unfortunately, since our attack relies on the resultant heuristic, it becomes a heuristic as well. But again in our experiments, we could not find a single failure of the resultant heuristic. The results are illustrated in Figure 6.2 in the same fashion as before.

Our new results raise the question whether it is possible to derive methods that work for all keys $e < \phi(N)$? In the light of our new bounds, this goal does not seem to be out of reach. Maybe a modification of our lattices could already suffice (e.g. using non-triangular lattice bases, see Chapter 7).

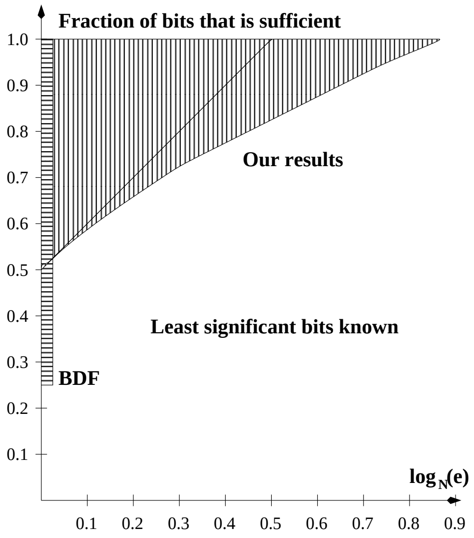

Figure 6.2: The results for known LSBs of d.

Known bits in CRT-variants:

In addition to the previously mentioned results, we provide results for attacking RSA when the Chinese Remainder Theorem (CRT) is used for the decryption process. As mentioned in Chapter 2, it is common practice to use the Quisquater-Couvreur method in order to speed up RSA decryption by a factor of approximately four. There also exist other fast CRT-RSA variants that make use of the values $d_p := d \bmod p-1$ and $d_q := d \bmod q-1$ like for instance Takagi's scheme, which additionally uses a modulus of the form $p^r q$ (see also Section 6.6).

Fast CRT-RSA variant are especially interesting for time-critical applications like smart-cards. On the other hand, it is well-known that smart-cards are vulnerable to many side-channel attacks, since an attacker who is in possession of the smart-card has many opportunities to manipulate the computations. However, it has never been studied in literature how many bits of d_p (or symmetrically of d_q) suffice to find the factorization of N.

Goal 2: Find the factorization of N using only a fraction of the bits of d_p.

We provide provable attacks for both cases: LSBs and MSBs. Interestingly, in our proofs we use the more general variant of Coppersmith's Theorem for MSBs of p (Theorem 10) that was introduced in Section 3.2: The knowledge of an approximation of kp for some (unknown) k up to an additive error of $N^{\frac{1}{4}}$ suffices to factor N in polynomial time.

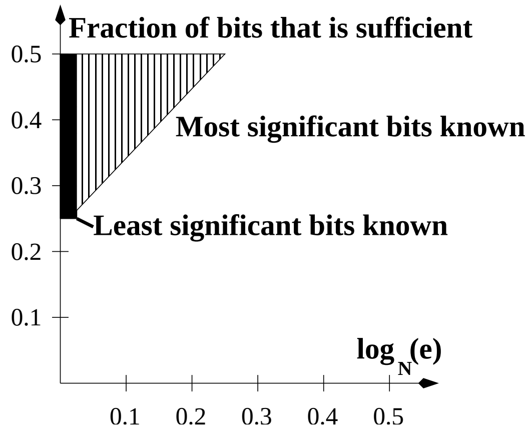

Figure 6.3: The results for known LSBs/MSBs of d_p.

We prove that for low public exponents e (i.e., $e = \text{poly}(\log N)$), half of the LSBs of d_p always suffice to factor N. Therefore, the attack is a threat to RSA implementations with the commonly used public exponents $e = 3$ and $e = 2^{16} + 1$. Note that half of the bits of d_p is only an amount of a quarter of the bits of N and thus the result is as strong as the best known partial key exposure attacks.

In the case of known MSBs of d_p, we present an algorithm that even works for all $e < N^{\frac{1}{4}}$ in polynomial time. Again for low public exponent RSA, it requires only half of the MSBs of d_p in order to factor N. The fraction of bits of d_p that is sufficient for the attack increases linearly with the bit-size of e. The results are illustrated in Figure 6.3.

Detailed overview

We briefly overview all known polynomial time partial key exposure attack in Figure 6.4 by giving the precise functions of the bits that have to be known. Here, we denote by $\alpha := \log_N(e)$ the size of e in terms of N.

It is worth noticing, that the new partial key exposure attacks on RSA-variants with CRT are the first rigorous methods that do not require any further restrictions.

Extensions to moduli of the form $N = p^r q$

We also investigate public moduli $N = p^r q$ for some constant $r > 1$. Moduli of this form have recently been used in different cryptographic designs. Fujioke, Okamoto and Uchiyama [30] presented an electronic cash scheme using a modulus $N = p^2 q$. Okamoto and Uchiyama [54] further designed an elegant public key cryptosystem that is provably as secure as factoring a modulus $N = p^2 q$.

A very fast CRT-RSA variant using moduli of the form $N = p^r q$ was introduced by Takagi [67] in 1998. The larger one chooses r, the more efficient is Takagi's scheme. On

	$\alpha := \log_{\mathbf{N}}(\mathbf{e})$	Fraction of bits that is needed	Known bits	Restriction
Theorem 51,1.&2.	$[\frac{1}{4}, \frac{1}{2}]$	α	MSBs of d	e prime/known fact.
Theorem 51,3.	$[0, \frac{1}{2}]$	$1-\alpha$	MSBs of d	$\frac{d}{\phi(N)} = \Omega(1)$
Section 6.2	$[\frac{1}{2}, \frac{\sqrt{6}-1}{2}]$	$1 - \frac{1}{8}\left(5 - 2\alpha - \sqrt{36\alpha^2 + 12\alpha - 15}\right)$	MSBs of d	heuristic
Theorem 51,4.	$[0, \frac{1}{2}]$	$\frac{3}{4}$	MSBs of d	$\frac{d}{\phi(N)}, \frac{\lvert p-q \rvert}{\sqrt{N}} = \Omega(1)$
Section 6.5	$[0, \frac{1}{4}]$	$\frac{1}{4} + \alpha$	MSBs of d_p	
Theorem 50	$\mathcal{O}(\log_N \log N)$	$\frac{1}{4}$	LSBs of d	$N = 3 \bmod 4$
Section 6.3	$[0, \frac{1}{2}]$	$\frac{1}{2} + \alpha$	LSBs of d	all but $\mathcal{O}(N^{\alpha-\epsilon})$ e's
Section 6.4	$[0, \frac{7}{8}]$	$\frac{1}{6} + \frac{1}{3}\sqrt{1+6\alpha}$	LSBs of d	heuristic
Section 6.5	$\mathcal{O}(\log_N \log N)$	$\frac{1}{4}$	LSBs of d_p	

Figure 6.4: Detailed summary of the results

the other hand, Boneh, Durfee and Howgrave-Graham [67] showed in 1999 that moduli of the form $N = p^r q$ are more susceptible to attacks that leak bits of p. Generalizing Coppersmith's result, they showed that it suffices to know a fraction of $\frac{1}{r+1}$ of the MSBs of p to factor the modulus (see Chapter 3, Theorem 13). We study how many bits of d (respectively of d_p) suffice in order to find the factorization of N in polynomial time.

Goal 3: Find the factorization of $N = p^r q$ using only a fraction of the bits of d (respectively the bits of d_p).

Takagi [67] extended Wiener's attack for low secret exponents d and showed a bound of $d < N^{\frac{1}{2(r+1)}}$. We propose two new rigorous methods based on Coppersmith's method for finding small roots of univariate modular polynomial equations. The new approaches improve Takagi's bound to

$$d < N^{\frac{r}{(r+1)^2}} \quad \text{or} \quad d < N^{\left(\frac{r-1}{r+1}\right)^2}$$

for $r \geq 2$. The first new bound is the best bound for the choice $r = 2$, whereas the second new bound improves upon the previous ones for $r \geq 3$. Note that the last bound tends to N for $r \to \infty$. Therefore the larger r is chosen, the fewer bits of d have to be known by an attacker.

Both new attacks possess interesting properties which the Wiener attack and the Boneh-Durfee attack do not share:

- One cannot counteract the attacks by choosing large public exponents e.

- The attacks immediately yield partial key exposure attacks. Namely, it makes no difference in the attacks whether the MSBs of d are zero (and thus d is a small decryption exponent) or are known to the adversary. In addition, the new partial key exposure attacks also do not depend on the size of e. These are the first attacks that work for arbitrary e.

Using the first attack, we are able to prove that a fraction of

$$1 - \frac{r}{(r+1)^2} \text{ of the MSBs or LSBs of } d$$

suffice to find the factorization of $N = p^r q$. The second attack yields partial key exposure attacks that require only a fraction of

$$\frac{4r}{(r+1)^2} \text{ of the MSBs or LSBs of } d$$

in order to factor N.

Since many schemes — such as for instance Takagi's scheme — with moduli $N = p^r q$ do not directly use the decryption exponent d but instead the value $d_p := d \bmod (p-1)$, partial key exposure attacks that use bits of d cannot be applied. Therefore, it is an interesting task to derive partial key exposure attacks for known bits of d_p, too.

In this case, we are able to generalize our partial key exposure attacks for d_p with moduli $N = pq$ (see Section 6.5) to the general case $N = p^r q$. Interestingly, the results are again much better for $r > 1$. Namely, we show that there exists a rigorous attack for known LSBs of d_p that needs a fraction of

$$\frac{1}{r+1}$$

of the bits of d_p.

Our new results show that moduli of the form $N = p^r q$ are more susceptible to attacks that leak bits of the secret exponent than the original RSA scheme.

A note on the presentation of the results

We introduced the topic of partial key exposure attacks by assuming that an attacker gets possession of a fraction of the secret key bits. This is a reasonable scenario coming from various side-channel attacks. However, we point out that our new results as well as the results of Boneh, Durfee and Frankel do not only cover the case of known bits but are slightly more general. To be more precise: When we talk of k known LSBs of d, then in fact an attacker needs to know integers d_0, M such that $d_0 = d \bmod M$, where $M \geq 2^k$. Thus, $M = 2^k$ is only the special case where he really knows the bits. Although we will talk of known LSBs in the following chapters, we find it convenient to state our results in the more general form using the parameters d_0 and M. There might be situations, where an RSA-variant does not leak the bits of d itself but the value $d_0 = d \bmod M$ for some M. We present our results in a form such that they can be applied to this case as well.

An analogous reasoning holds for the case of MSBs. An attacker does not really have to know the MSBs of d but instead an approximation $\tilde{d}$ of d such that $|d - \tilde{d}|$ can be suitably upper-bounded.

Throughout this chapter, we will assume that an RSA modulus $N = pq$ is a product of two primes of the same bit-size. From Section 2.1 we know that in this case

$$p + q \leq 3\sqrt{N}.$$

6.2 MSBs known: A method for $e \in [N^{\frac{1}{2}}, N^{\frac{\sqrt{6}-1}{2}})$

In this section, we present a polynomial time attack on RSA for public exponents e in the interval $[N^{\frac{1}{2}}, N^{\frac{\sqrt{6}-1}{2}})$ given most significant bits of d. This answers an open question of Boneh, Durfee and Frankel whether there exist partial key exposure attacks beyond the bound $e = \sqrt{N}$.

Our approach makes use of Coppersmith's method for modular polynomial equations in the trivariate case, which is a heuristic (see Section 3.4). The only heuristic part in this method as well as in our attack is the resultant heuristic. The method fails iff one of the resultant computations yields the zero polynomial. In order to state our results as a theorem, we assume that this failure never happens.

Assumption 52 *The resultant computations for the polynomials constructed in this section yield non-zero polynomials.*

We made various experiments to support this assumption, and we never found a counter-example.

Theorem 53 *Under Assumption 52, for every fixed $\epsilon > 0$ there exists an integer N_0 such that for every $N > N_0$ the following holds:*
Let (N, e) be an RSA public key, where $\alpha := \log_N(e)$ is in the range $[\frac{1}{2}, \frac{\sqrt{6}-1}{2}]$. Suppose we are given an approximation $\tilde{d}$ of d with

$$|d - \tilde{d}| \leq N^{\frac{1}{8}\left(5 - 2\alpha - \sqrt{36\alpha^2 + 12\alpha - 15}\right) - \epsilon}.$$

Then N can be factored in time polynomial in $\log N$.

Before we start to prove Theorem 53, we want to provide some experimental results in Figure 6.5 to give an impression of the amount of bits that is needed in our partial key exposure attack (see also Figure 6.1). The experiments confirm the reliability of the multivariate heuristic and support our Assumption 52.

Define $\delta := \frac{1}{8}\left(5 - 2\alpha - \sqrt{36\alpha^2 + 12\alpha - 15}\right) - \epsilon$. Then a fraction of $1 - \delta$ of the MSBs of d is required (asymptotically) for the new attack. For $\alpha = 0.55$ this is a 0.71-fraction and for $\alpha = 0.6$ we need a 0.81-fraction. The experiments shown in Figure 6.5 confirm that one can get close to these bounds in practice by spending a reasonable amount of computing time. All of our experiments were carried out on a 500-MHz workstation using Shoup's NTL [63].

N	e	**known MSBs**	Lattice parameters	L^3-time
1000 bit	*600 bit*	**955 bit**	$m = t = 1,\ \dim(L) = 7$	1 sec
1000 bit	550 bit	**855 bit**	$m = t = 1,\ \dim(L) = 7$	1 sec
1000 bit	*600 bit*	**905 bit**	$m = t = 2,\ \dim(L) = 19$	40 sec
1000 bit	550 bit	**810 bit**	$m = t = 2,\ \dim(L) = 19$	40 sec
1000 bit	*600 bit*	**880 bit**	$m = t = 3,\ \dim(L) = 50$	57 min
1000 bit	550 bit	**785 bit**	$m = t = 3,\ \dim(L) = 50$	72 min

Figure 6.5: Experimental results for known MSBs

Proof. [Theorem 53]: We start by looking at the public key equation

$$ed - 1 = k\phi(N), \qquad \text{where } k \in \mathbb{Z}. \tag{6.1}$$

Boneh, Durfee and Frankel [14] observed that a suitable fraction of the MSBs of d yields the parameter k. The main drawback of the methods presented in [14] is that they all require that k is known exactly. This restricts the methods' usability to public exponents $e \leq \sqrt{N}$.

Roadmap of the proof

- We relax the restriction of Boneh, Durfee and Frankel and show that an approximation of d yields an approximation $\tilde{k}$ of k.
- We use the approximation $\tilde{k}$ to construct a trivariate polynomial f_N with a small root (x_0, y_0, z_0) modulo N.
- Using Coppersmith's method we compute three polynomials with the root (x_0, y_0, z_0) over the integers.
- We compute (x_0, y_0, z_0) over the integers using resultant computations. Under Assumption 52, this will lead to the factorization of N.

Let us look at the case where we obtain only an approximation $\tilde{k}$ of k from an approximation $\tilde{d}$ of d. Define $\tilde{k} := \frac{e\tilde{d}-1}{N+1}$, then

$$|k - \tilde{k}| = \left| \frac{ed-1}{\phi(N)} - \frac{e\tilde{d}-1}{N+1} \right|.$$

Taking the common divisor on the right-hand side gives us:

$$\left| \frac{(ed-1)(N+1) - (e\tilde{d}-1)(N+1-(p+q))}{\phi(N)(N+1)} \right|.$$

Now, we can estimate

$$|k - \tilde{k}| \leq \left| \frac{e(d-\tilde{d})}{\phi(N)} \right| + \left| \frac{(p+q)(e\tilde{d}-1)}{\phi(N)(N+1)} \right| \leq \frac{e}{\phi(N)}(N^{\delta} + 3N^{-\frac{1}{2}}\tilde{d}).$$

We claim that the hard case is the one where the term $N^{-\frac{1}{2}}\tilde{d}$ dominates N^{δ}. Let us first assume the opposite, i.e., $N^{\delta} > N^{-\frac{1}{2}}\tilde{d}$. In this case, $|k - \tilde{k}|$ can be bounded by $N^{\alpha+\delta-1}$, where we neglect low order terms. Hence whenever $\alpha + \delta - 1 \leq 0$, then k can be determined exactly. Note that the condition in Theorem 53 implies the desired inequality $\delta \leq 1 - \alpha$.

But if k is known, we can compute $p + q = N + 1 + k^{-1} \bmod e$. On the other hand $e \geq N^{\frac{1}{2}}$ and therefore we get $p + q$ over the integers and not just modulo e. This leads to the factorization of N. Hence we assume in the following that we are in the case $N^{-\frac{1}{2}}\tilde{d} \geq N^{\delta}$. In this case we can bound $|k - \tilde{k}|$ by $4N^{\alpha-\frac{1}{2}}$.

Now let us define $d_0 := d - \tilde{d}$ and $k_0 := k - \tilde{k}$. Then we can reformulate equation (6.1) as

$$e(\tilde{d} + d_0) - 1 = (\tilde{k} + k_0)\phi(N).$$

This can also be written as

$$ed_0 + (\tilde{k} + k_0)(p + q - 1) + e\tilde{d} - 1 = (\tilde{k} + k_0)N. \tag{6.2}$$

Equation (6.2) gives us a trivariate polynomial

$$f_N(x, y, z) = ex + (\tilde{k} + y)z + e\tilde{d} - 1$$

with the root $(x_0, y_0, z_0) = (d_0, k_0, p + q - 1)$ modulo N. Define the upper bounds $X := N^\delta$, $Y := 4N^{\alpha - \frac{1}{2}}$ and $Z := 3N^{\frac{1}{2}}$. Then we have $x_0 \leq X$, $y_0 \leq Y$ and $z_0 \leq Z$.

Now we use Coppersmith's method in order to construct from $f_N(x, y, z)$ a polynomial $f(x, y, z)$ with the same root (x_0, y_0, z_0) over $\mathbb{Z}$ (and not just modulo N). Let us recall Howgrave-Graham's theorem (Theorem 14) in the trivariate case:

Theorem 54 (Howgrave-Graham) *Let $f(x, y, z)$ be a polynomial that is a sum of at most n monomials. Suppose that*

(1) $f(x_0, y_0, z_0) = 0 \bmod N^m$, where $|x_0| \leq X$, $|y_0| \leq Y$ and $|z_0| \leq Z$

(2) $\|f(xX, yY, zZ)\| < \frac{N^m}{\sqrt{n}}$.

Then $f(x_0, y_0, z_0) = 0$ holds over the integers.

We construct polynomials that all satisfy condition *(1)* of Howgrave-Graham's Theorem. Thus, every integer linear combination of these polynomials also satisfies the first condition. We search among these linear combinations for a polynomial f that satisfies condition *(2)*. This will be done using the L^3-lattice reduction algorithm.

Let us start by defining the following polynomials $g_{i,j,k}(x, y, z)$ and $h_{i,j,k}(x, y, z)$ for some fixed integers m and t:

$$\begin{aligned} g_{i,j,k} &:= x^{j-k} z^k N^i f_N^{m-i} \quad \text{for } i = 0, \ldots, m;\ j = 0, \ldots, i;\ k = 0, \ldots, j \\ h_{i,j,k} &:= x^j y^k N^i f_N^{m-i} \quad \text{for } i = 0, \ldots, m;\ j = 0, \ldots, i;\ k = 1, \ldots, t \end{aligned}$$

The parameter t has to be optimized as a function of m.

One can build a lattice $L(m)$ by using the coefficient vectors of the polynomials $g_{i,j,k}(xX, yY, zZ)$ and $h_{i,j,k}(xX, yY, zZ)$ as basis vectors for a basis $B(m)$ of $L(m)$. We provide an example for the case $m = 1$ and $t = 1$, where the coefficient vectors of $g_{1,0,0}$, $g_{1,1,0}$, $g_{1,1,1}$, $g_{0,0,0}$ and $h_{1,0,1}$, $h_{1,1,1}$, $h_{0,0,1}$ form (as row vectors) the following lattice basis $B(1)$.

$$B(1) = \left[\begin{array}{cccc|ccc} N & & & & & & \\ & NX & & & & & \\ & & NZ & & & & \\ e\tilde{d}-1 & eX & \tilde{k}Z & YZ & & & \\ \hline & & & & NY & & \\ & & & & & NXY & \\ & & & \tilde{k}YZ & (e\tilde{d}-1)Y & eXY & Y^2Z \end{array}\right]$$

The following lemma shows, that the L^3-algorithm always finds at least three different vectors in $L(m)$ that satisfy condition *(2)* of Howgrave-Graham's Theorem.

Lemma 55 *Let $X := N^{\delta}$, $Y := 4N^{\alpha-\frac{1}{2}}$ and $Z := 3N^{\frac{1}{2}}$. Then one can find three linearly independent vectors in $L(m)$ with norm smaller than $\frac{N^m}{\sqrt{\dim L(m)}}$ using the L^3-algorithm.*

Proof: Let $n := \dim L(M)$ denote the lattice dimension. We want to find a reduced basis of $L(m)$ with three basis vectors smaller than $\frac{N^m}{\sqrt{n}}$. Applying Theorem 4, we know that for an L^3-reduced basis $\{v_1', v_2', \ldots, v_n'\}$

$$\|v_1'\| \leq \|v_2'\| \leq \|v_3'\| \leq 2^{\frac{n(n-1)}{4(n-2)}} \det L(M)^{\frac{1}{n-2}}.$$

Since we need $\|v_3'\| < \frac{N^m}{\sqrt{n}}$, we have to satisfy the condition

$$\det(L) < cN^{m(n-2)},$$

where $c = 2^{-\frac{n(n-1)}{4}} n^{-\frac{n-2}{2}}$ does not depend on N and therefore contributes to the error term ϵ.

Let $t := \tau m$, then the determinant of $L(M)$ is

$$\det L(M) = \left(N^{8\tau+3} X^{4\tau+1} Y^{6\tau^2+4\tau+1} Z^{4\tau+2}\right)^{\frac{1}{24}m^4(1+o(1))}.$$

Using the bounds $X = N^{\delta}$, $Y = 4N^{\alpha-\frac{1}{2}}$ and $Z = 3N^{\frac{1}{2}}$ we obtain

$$\det L(M) = N^{\frac{1}{24}m^4(3\tau^2(2\alpha-1)+4\tau(\delta+\alpha+2)+\delta+\alpha+\frac{7}{2})(1+o(1))}.$$

An easy calculation shows that $n = \frac{1}{24}m^3(12\tau+4)(1+o(1))$. Neglecting low order terms, our condition simplifies to

$$3\tau^2(2\alpha-1) + 4\tau(\delta+\alpha-1) + \delta + \alpha - \frac{1}{2} < 0.$$

The left hand side minimizes for the choice $\tau = \frac{2}{3}\frac{1-\delta-\alpha}{2\alpha-1}$. Plugging this value in, we obtain the desired condition

$$\delta \leq \frac{1}{8}\left(5 - 2\alpha - \sqrt{36\alpha^2 + 12\alpha - 15}\right)$$

This concludes the proof of the lemma. $\square$

Combining Theorem 54 and Lemma 55, we obtain from the three vectors with norm smaller than $\frac{N^m}{\sqrt{\dim L(m)}}$ three polynomials $f_1(x,y,z)$, $f_2(x,y,z)$ and $f_3(x,y,z)$ with the common root (x_0, y_0, z_0). Our goal is to extract the value $z_0 = p + q - 1$. The equation $N = pq$ together with the number z_0 yields the factorization of N.

Therefore, we take the resultants $\text{res}_x(f_1, f_2)$ and $\text{res}_x(f_1, f_3)$ with respect to x. The resulting polynomials g_1 and g_2 are bivariate polynomials in y and z. In order to remove the unknown y, we compute the resultant $\text{res}_y(g_1, g_2)$ which is an univariate polynomial in z. The root z_0 most be among the roots of this polynomial. Thus, if $\text{res}_y(g_1, g_2)$ is not the zero polynomial (Assumption 52 excludes this case) then z_0 can be found by standard root finding algorithms.

Let us briefly summarize the whole factorization algorithm.

Algorithm MSB-Attack

INPUT: – (N, e), where $N = pq$ and d satisfies $ed = 1 \bmod \phi(N)$

– $\tilde{d}$ with $|d - \tilde{d}| \leq N^{\frac{1}{8}\left(5-2\alpha-\sqrt{36\alpha^2+12\alpha-15}\right)-\epsilon}$, where $\alpha = \log_N(e)$.

1. Fix an integer m (depending on $\frac{1}{\epsilon}$) and construct the lattice $L(m)$ with basis $B(m)$.
2. Find three vectors v_1, v_2, v_3 with norm smaller than $\frac{N^m}{\sqrt{\dim L(m)}}$ using L^3-reduction.
3. Construct from v_1, v_2, v_3 three polynomials $f_1(x, y, z)$, $f_2(x, y, z)$ and $f_3(x, y, z)$ and compute the univariate polynomial $h(z) = \text{res}_y\left(\text{res}_x(f_1, f_2), \text{res}_x(f_1, f_3)\right)$.
4. Compute all the roots of $h(z)$. For every root z_0 test whether the solutions of the quadratic $x^2 - (z_0 + 1)x - N$ are the prime factors p, q.

OUTPUT: p, q

Since m is a constant, $L(m)$ has constant dimension and the entries in the basis matrix $B(m)$ have bit-size $\text{poly}(\log N)$. Therefore, the L^3-algorithm runs in time polynomial in $\log(N)$. Since v_1, v_2, v_3 have fixed dimension, the polynomial $h(z)$ has also fixed dimension and its coefficients are bounded by a polynomial in N. Thus the roots as well as the solutions of the quadratic are computable in polynomial time. This concludes the proof of Theorem 53. $\diamond$

6.3 LSBs known: A provable method for $e < N^{\frac{1}{2}}$

In this section, we present a provable attack on RSA with public exponent $e < N^{\frac{1}{2}}$, where we know $d_0 = d \bmod M$ for some modulus M. For instance, assume that an attacker succeeds to get the lower k bits of d, then $M = 2^k$.

In the following we show that whenever M is sufficiently large, then N can be factored in polynomial time for all but a negligible fraction of choices for e. As a by-product, our proof provides an elementary method how to prevent resultant computations when Coppersmith's method is applied to multivariate polynomials of a simple form.

Theorem 56 *Let N be an RSA modulus and let $0 < \alpha, \epsilon < \frac{1}{2}$. For all but a $\mathcal{O}(\frac{1}{N^\epsilon})$-fraction of the public exponents e in the interval $[3, N^\alpha]$ the following holds: Let d be the secret key. Suppose we are given d_0, M satisfying $d = d_0 \bmod M$ with*

$$N^{\alpha+\frac{1}{2}+\epsilon} \leq M \leq 2N^{\alpha+\frac{1}{2}+\epsilon}.$$

Then the factorization of N can be found in polynomial time.

Before we prove the theorem, we want to give some experimental results. We tested our algorithm with the commonly used public exponent $e = 2^{16}+1$ and varying 1000-bit moduli N, where we knew 525 LSBs of d. Note that in comparison to the Boneh-Durfee-Frankel-approach for LSBs (see Theorem 50), we need about twice as many bits but in their method one has to run about e times a lattice reduction. The running time of our algorithm is about 1 second on a 500 MHz workstation. In 100 experiments, the algorithm never failed to yield the factorization of N.

Proof. [Theorem 56] We start by looking at the RSA key equation $ed - 1 = k\phi(N)$. Let us write $d = d_1 M + d_0$, where d_1 is the unknown part of d. Then

$$ed_1 M + k(p+q-1) - 1 + ed_0 = kN. \tag{6.3}$$

Equation (6.3) in turn gives us a bivariate polynomial

$$f_N(x, y) = eMx + y + ed_0$$

with a root $(x_0, y_0) = (d_1, k(p+q-1)-1)$ modulo N.

Roadmap of the proof

- We show that Coppersmith's method can be used for all but a negligible fraction of the public exponents e to construct two *linear* polynomials $f_1(x, y)$ and $f_2(x, y)$ with the root (x_0, y_0) over $\mathbb{Z}$, whose coefficient vectors are linearly independent. In order to prove this, we have to show that a certain 3-dimensional lattice always contains two sufficiently short, linearly independent vectors.

- The common root (x_0, y_0) of f_1 and f_2 can then be computed using elementary Gaussian elimination. The term (x_0, y_0) leads to the factorization of N.

In order to bound y_0 notice that

$$k = \frac{ed-1}{\phi(N)} < e\frac{d}{\phi(N)} < e \le N^{\alpha}.$$

Since $d_1 \le \frac{N}{M}$, we can set the bounds $X := N^{\frac{1}{2}-\alpha-\epsilon}$ and $Y := 3N^{\frac{1}{2}+\alpha}$ satisfying $x_0 \le X$ and $y_0 \le Y$.

We want to transform our polynomial $f_N(x, y)$ into two polynomials with the root (x_0, y_0) over the integers. Therefore, we apply Howgrave-Graham's Theorem (Theorem 14) in the bivariate case. For this purpose we take the auxiliary polynomials N and Nx which are both the zero polynomial modulo N. Thus, every integer linear combination $f = a_0N + a_1Nx + a_2f_N(x, y)$ has the root (x_0, y_0) modulo N.

According to the second condition of Howgrave-Graham's theorem we have to look for integer linear combinations f satisfying $\|f(xX, yY)\| \le \frac{N}{\sqrt{3}}$. Thus, we search for suitably small vectors in the lattice L given by the span of the row vectors of the following (3×3)-lattice base

$$B = \begin{bmatrix} N & & \\ & NX & \\ ed_0 & eMX & Y \end{bmatrix}.$$

Now, our goal is to find two linearly independent vectors $(a_0, a_1, a_2)B$ and $(b_0, b_1, b_2)B$ both having norm smaller than $\frac{N}{\sqrt{3}}$. Since L has dimension 3, we can compute two shortest linearly independent vectors in L in polynomial time using an algorithm of Blömer [4]. In practice, the L^3-algorithm will suffice.

Assume we could find two linearly independent vectors with norm smaller than $\frac{N}{\sqrt{3}}$. Then we obtain from Theorem 14 the following two equations

$$\begin{aligned} a_0N + a_1Nx_0 + a_2f_N(x_0, y_0) &= 0 \text{ and} \\ b_0N + b_1Nx_0 + b_2f_N(x_0, y_0) &= 0. \end{aligned}$$

From equation (6.3) we know that $f_N(x_0, y_0) = kN$. Hence, our equations simplify to the linear system

$$\begin{aligned} a_1x_0 + a_2k &= -a_0 \\ b_1x_0 + b_2k &= -b_0 \end{aligned} \tag{6.4}$$

Since $(a_0, a_1, a_2), (b_0, b_1, b_2) \in \mathbb{Z}^3$ are linearly independent and satisfy (6.4), we claim that the 2-dimensional vectors $(a_1, a_2), (b_1, b_2)$ are also linearly independent. Assume for contradiction that $c \cdot (a_1, a_2) = (b_1, b_2)$ holds for some $c \in \mathbb{Q}$. Then

$$c \cdot (a_1x_0, a_2k) = (b_1x_0, b_2k) \;\Rightarrow\; c \cdot (a_1x_0 + a_2k) = b_1x_0 + b_2k,$$

which implies that $c \cdot a_0 = b_0$. This leads to $c \cdot (a_0, a_1, a_2) = (b_0, b_1, b_2)$ contradicting the assumption that the vectors are linearly independent.

The linear independence of (a_1, a_2) and (b_1, b_2) ensures that we can determine x_0, k as the unique solution of the linear system (6.4) by using Gaussian elimination. Afterwards, we can derive y_0 by $y_0 = kN - eMx_0 - ed_0$. Therefore, $\frac{y_0+1}{k} = p + q - 1$ gives us the necessary term to factor the modulus N.

It remains to show that L contains indeed two linearly independent vectors with norm smaller than $\frac{N}{\sqrt{3}}$. The following lemma proves that this is satisfied for most choices of e using a counting argument.

Lemma 57 *Given N, α, ϵ and M as defined in Theorem 56. Then for all but $\mathcal{O}(N^{\alpha-\epsilon})$ choices of e in the interval $[3, N^{\alpha}]$ the following holds: Let $X := N^{\frac{1}{2}-\alpha-\epsilon}$ and $Y := 3N^{\frac{1}{2}+\alpha}$. Then the lattice L contains two linearly independent vectors with norm less than $\frac{N}{\sqrt{3}}$.*

Proof: Let us first define the notion of successive minima (see for instance [58]). The i^{th} successive minimum λ_i of a lattice L is

$$\lambda_i := \min \{r \mid \text{There exist } i \text{ linearly independent vectors in } L \text{ with norm at most } r\}.$$

We have to show that for most choices of e the second successive minima λ_2 of L is strictly less than $\frac{N}{\sqrt{3}}$. By Minkowski's second theorem (see [33]), we know that for any 3-dimensional lattice L and its successive minima $\lambda_1, \lambda_2, \lambda_3$

$$\lambda_1 \lambda_2 \lambda_3 \leq 2 \det(L).$$

In our case $\det(L) = N^2XY$. Hence for all e such that $\lambda_1 > 6XY$, we get $\lambda_2 < \frac{N}{\sqrt{3}}$ and we are done.

Now assume $\lambda_1 \leq 6XY$. Hence, we can find $c_0, c_1, c_2 \in \mathbb{Z}$ such that $\|(c_0, c_1, c_2)B\| < 6XY$. This implies

$$\begin{aligned} |c_2| &\leq 6X \quad \text{and} \\ \left| \frac{c_1}{c_2} + \frac{eM}{N} \right| &\leq \frac{6Y}{c_2 N}. \end{aligned}$$

Using $XY \leq 3N^{1-\epsilon}$, the second inequality implies

$$\left|\frac{c_1}{c_2} + \frac{eM}{N}\right| \leq \frac{18}{c_2 X N^{\epsilon}} \tag{6.5}$$

Next we bound the number of e's in $[3, N^{\alpha}]$ that can satisfy (6.5) for some ratio $\frac{c_1}{c_2}$.

Since e is positive, we can assume that $c_1 < 0$ and $c_2 > 0$ without loss of generality. Now we make the following series of observations.

- The difference between any two numbers of the form $\frac{eM}{N}$ is at least $\frac{M}{N} \geq N^{\alpha-\frac{1}{2}+\epsilon}$.
- If (6.5) is true for some ratio $\frac{c_1}{c_2}$ and some e then $\frac{eM}{N}$ must lie in the interval $\left[\frac{c_1}{c_2} - \frac{18}{c_2 X N^{\epsilon}}, \frac{c_1}{c_2} + \frac{18}{c_2 X N^{\epsilon}}\right]$.
- Combining the first two observations we conclude that for a fixed ratio $\frac{c_1}{c_2}$ there are at most $\frac{36}{c_2 X N^{\alpha-\frac{1}{2}+2\epsilon}}$ public keys e such that (6.5) is satisfied.
- Since $e \leq N^{\alpha}$ and $M \leq 2N^{2\alpha-\frac{1}{2}+\epsilon}$, we get $\frac{eM}{N} \leq 2N^{2\alpha-\frac{1}{2}+\epsilon}$. Consider a fixed but arbitrary c_2. Then (6.5) is satisfied for some c_1 and some public key e only if $c_1 \in [-2N^{2\alpha-\frac{1}{2}+\epsilon}c_2, -1]$.
- The previous two observations imply that for fixed c_2 the number of e's satisfying (6.5) is bounded by $\frac{72N^{\alpha-\epsilon}}{X}$.
- The previous observation and $c_2 \leq 6X$ imply, that the number of public keys e for which (6.5) is satisfied for some ratio $\frac{c_1}{c_2}$ is bounded by $432N^{\alpha-\epsilon}$.

The last observation concludes the proof of Lemma 57. $\square$

Let us summarize the attack.

Algorithm Rigorous LSB-Attack

INPUT: – (N, e), where $N = pq$ and d satisfies $ed = 1 \bmod \phi(N)$

– d_0, M with $d_0 = d \bmod M$ and $N^{\alpha+\frac{1}{2}+\epsilon} \leq M \leq 2N^{\alpha+\frac{1}{2}+\epsilon}$, where $\alpha = \log_N(e)$.

1. Construct the 3-dimensional lattice basis B of L and find two shortest, linearly independent vectors $v_1 = (a_0, a_1, a_2)B, v_2 = (b_0, b_1, b_2)B \in L$ using Blömer's algorithm.
2. Let $\|v_1\| \leq \|v_2\|$. If $\|v_2\| \geq \frac{N}{\sqrt{3}}$, then return "Failed".
3. Solve the linear system

$$\begin{array}{rcl} a_1x_0 + a_2k & = & -a_0 \\ b_1x_0 + b_2k & = & -b_0 \end{array}$$

in the unknown parameters x_0, k using Gaussian elimination.

4. Compute $y_0 = kN - eMx_0 - ed_0$.
5. Output the two solutions of the quadratic equation $x^2 - (\frac{y_0+1}{k} + 1)x - N$.

OUTPUT: p, q

Every step of the algorithm can be done in polynomial time. This concludes the proof of Theorem 56. $\diamond$

6.4 LSBs known: A method for all e with $e < N^{\frac{7}{8}}$

In this section, we improve the approach of Section 6.3 by taking multi-dimensional lattices. In contrast to Section 6.3 our results are not rigorous. They rely on Coppersmith's resultant heuristic for multivariate modular equations. As in Section 6.2, we make the assumption that the only heuristic part of the computations in our method — namely one resultant calculation — never leads to a failure of the approach.

Assumption 58 *The resultant computation for the two polynomials constructed in the method of this section yields a non-zero polynomial.*

We confirmed the reliability of Assumption 58 by various experiments: None of our experiments has ever failed to yield the factorization of N.

Interestingly, the bound that we get in this section is even better than the bound derived for known MSBs (see Section 6.2). Namely, we obtain an attack that works for all $e < N^{\frac{7}{8}}$.

Theorem 59 *Under Assumption 58, for every $\epsilon > 0$ there exists N_0 such that for every $N \geq N_0$ the following holds:*
Let (N, e) be an RSA public key with $\alpha := \log_N(e) \leq \frac{7}{8}$. Let d be the secret key. Suppose we are given d_0, M satisfying $d = d_0 \bmod M$ with

$$M \geq N^{\frac{1}{6}+\frac{1}{3}\sqrt{1+6\alpha}+\epsilon}.$$

Then N can be factored in polynomial time.

Before we start with the proof of Theorem 59, we provide some experimental results in Figure 6.6 to given an impression of the number of bits that are sufficient in our partial key exposure attack (see also Figure 6.2).

N	e	**known LSBs**	Lattice parameters	L^3-time
1000 bit	300 bit	**805 bit**	$m = 1, t = 0,\ \dim(L) = 3$	1 sec
1000 bit	300 bit	**765 bit**	$m = 7, t = 1,\ \dim(L) = 44$	405 min
1000 bit	400 bit	**880 bit**	$m = 3, t = 1,\ \dim(L) = 14$	40 sec
1000 bit	400 bit	**840 bit**	$m = 6, t = 1,\ \dim(L) = 35$	196 min
1000 bit	500 bit	**920 bit**	$m = 4, t = 1,\ \dim(L) = 20$	7 min
1000 bit	500 bit	**890 bit**	$m = 8, t = 2,\ \dim(L) = 63$	50 hours

Figure 6.6: Experimental results for known LSBs

Proof. [Theorem 59] We start by looking at the equation $ed - 1 = k\phi(N)$. As in Section 6.3, we write $d = d_1 M + d_0$. This gives us the equation

$$k(N - (p + q - 1)) - ed_0 + 1 = eMd_1. \tag{6.6}$$

From (6.6) we obtain the bivariate polynomial

$$f_{eM}(y,z) = y(N-z) - ed_0 + 1$$

with the root $(y_0, z_0) = (k, p+q-1)$ modulo eM. Analogous to Section 6.3 we can derive the bounds $Y := N^{\alpha}$ and $Z := 3N^{\frac{1}{2}}$ satisfying $y_0 \le Y$ and $z_0 \le Z$.
Fix some integers m and t. Define the polynomials

$$\begin{array}{lcll} g_{i,j} & := & y^j (eM)^i f_{eM}^{m-i} & \text{for } i = 0,\ldots,m;\ j = 0,\ldots,i \\ h_{i,j} & := & z^j (eM)^i f_{eM}^{m-i} & \text{for } i = 0,\ldots,m;\ j = 1,\ldots,t. \end{array}$$

The parameter t has to be optimized as a function of m.

Since all the polynomials have a term $(eM)^i f_{eM}^{m-i}$, all integer linear combinations of the polynomials have the root (y_0, z_0) modulo $(eM)^m$, i.e., they satisfy the first condition of Howgrave-Graham's theorem (Theorem 14 in the bivariate case). Let $L(m)$ be the lattice defined by the basis $B(m)$, where the coefficient vectors of $g_{i,j}(yY, zZ)$ and $h_{i,j}(yY, zZ)$ are the basis vectors of $B(m)$ (with the same parameter choices of i and j as before).

In order to fulfill the second condition in Howgrave-Graham's theorem, we have to find a vector in $L(m)$ with norm less than $\frac{(eM)^m}{\sqrt{\dim L(m)}}$.

The following lemma shows that one can always find two sufficiently short vectors in $L(m)$ using the L^3-algorithm.

Lemma 60 *Let e, M be as defined in Theorem 59. Suppose $Y := N^{\alpha}$ and $Z := 3N^{\frac{1}{2}}$. Then the L^3-algorithm finds at least two vectors in $L(M)$ with norm smaller than $\frac{(eM)^m}{\sqrt{\dim L(m)}}$.*

Proof: Set $n = \dim L(M)$. Applying Theorem 4, we know that the second-to-shortest vector v_2' of an L_3-reduced base satisfies $\|v_2'\| \le 2^{\frac{n}{4}} \det L(M)^{\frac{1}{n-1}}$. Thus, we have to satisfy the condition

$$2^{\frac{n}{4}} \det L(M)^{\frac{1}{n-1}} < \frac{(eM)^m}{\sqrt{n}}.$$

Neglecting all terms that do not depend on N, the condition simplifies to $\det L(M) < (eM)^{m(n-1)}$. We set $t = \tau m$. Then, a straightforward calculation shows that

$$\det L(M) = \left((eMX)^{3\tau+2} Z^{3\tau^2+3\tau+1}\right)^{\frac{1}{6}m^3(1+o(1))}.$$

If we plug in the bounds $Y = N^{\alpha}$ and $Z = 3N^{\frac{1}{2}}$, we obtain the new condition

$$N^{\frac{1}{12}m^3(3\tau^2+3\tau(2\alpha+1)+4\alpha+1)(1+o(1))} \le (eM)^{m(n-1)-(3\tau+2)(\frac{1}{6}m^3(1+o(1)))}.$$

On the other hand, we know that $eM \geq N^{\alpha+\frac{1}{6}+\frac{1}{3}\sqrt{1+6\alpha}+\epsilon}$. An easy computation shows that $n = (\tau + \frac{1}{2})m^2$. Again neglecting all low order terms, we obtain the new condition

$$9\tau^2 + 6(\alpha + \tau) - 2\sqrt{1+6\alpha}(1+3\tau) + 2 \leq 0$$

The left-hand side minimizes for the parameter choice $\tau = \frac{1}{3}(\sqrt{1+6\alpha} - 1)$. For this setting of τ, our condition is satisfied. This concludes the proof of Lemma 60. $\Box$

Combining Theorem 14 and Lemma 60, we obtain two polynomials $f_1(y,z)$, $f_2(y,z)$ with the common root (y_0, z_0) over the integers. By Assumption 58, the resultant $\mathrm{res}_y(f_1, f_2)$ is not zero such that we can find $z_0 = p + q - 1$ using standard root finding algorithms. This gives us the factorization of N.

We briefly summarize the algorithm of the attack.

Algorithm Heuristic LSB-Attack

INPUT: – (N, e), where $N = pq$ and d satisfies $ed = 1 \bmod \phi(N)$

– d_0, M with $d_0 = d \bmod M$ and $M \geq N^{\frac{1}{6}+\frac{1}{3}\sqrt{1+6\alpha}+\epsilon}$, where $\alpha = \log_N(e)$.

1. Fix an integer m (depending on $\frac{1}{\epsilon}$) and construct the basis $B(m)$ of the lattice $L(m)$.
2. Find two vectors $v_1, v_2 \in L(m)$ with norm less than $\frac{(eM)^m}{\sqrt{\dim L(m)}}$ in $L(M)$ using the L^3-algorithm.
3. Construct two bivariate polynomial $f_1(y,z)$, $f_2(y,z)$ from v_1, v_2 and compute $g(z) = \mathrm{res}_x(f_1, f_2)$.
4. For every root z_0 of $g(z)$, test whether the solutions of the quadratic equation $x^2 - (z_0+1)x - N$ equal the factors p, q.

OUTPUT: p, q

Since m is fixed, $L(m)$ has constant dimension and the basis matrix $B(m)$ has entries that are bounded by a polynomial in N. This implies that the L^3-algorithm runs in polynomial time. Additionally, $g(z)$ has constant degree and coefficients bounded by $\mathrm{poly}(N)$. Thus, every step of the algorithm can be done in time polynomial in $\log(N)$. This concludes the proof of Theorem 59. $\diamond$

6.5 Known MSBs/LSBs and Chinese Remaindering

In order to speed up the decryption/signing process, it is common practice to use the values $d_p := d \bmod p-1$ and $d_q := d \bmod q-1$ (see the Quisquater-Couvreur method in Section 2.1).

Fast RSA decryption/signature variants are especially interesting for time-critical applications like smart-cards, which in turn are highly vulnerable to side-channel attacks. Therefore, it is interesting to study how many bits of d_p (or symmetrically of d_q) suffice in order to find the factorization of N. We present two rigorous results for both cases — LSBs and MSBs — in this section (see also Figure 6.3).

Both of our proofs use Theorem 10, which in turn is based on Coppersmith's method in the univariate case. We recall this theorem here.

Theorem 10 (Coppersmith: MSBs of kp) *Let $N = pq$ with $p > q$. Furthermore, let k be an (unknown) integer that is not a multiple of q. Suppose we know an approximation $\tilde{p}$ of kp with*

$$|\, kp - \tilde{p}\,| \le 2N^{\frac{1}{4}}.$$

Then we can find the factorization of N in time polynomial in $\log N$.

First, we consider the case of known LSBs of d_p. We show that whenever the public exponent e is of size poly$(\log N)$, then half of the lower bits of d_p are sufficient to find the factorization of N in polynomial time.

Throughout this section, we assume wlog that $p > q$. Since p and q are of the same bit-size, we know that $p < 2\sqrt{N}$.

Theorem 61 *Let (N, e) be an RSA public key with $N = pq$ and secret key d. Let $d_p := d \bmod p-1$. Suppose we are given d_0, M with $d_0 := d_p \bmod M$ and*

$$M \ge N^{\frac{1}{4}}.$$

Then the factorization of N can be found in time $e \cdot poly(\log N)$.

Proof: We know that

$$ed_p - 1 = k(p-1)$$

for some $k \in \mathbb{Z}$. Since $d_p < p-1$, we know that $k = \frac{ed_p-1}{p-1} < e$. Let us write $d_p = d_1 M + d_0$, where $d_1 < \frac{d_p}{M} < \frac{p}{M} \le 2N^{\frac{1}{4}}$. We can rewrite our equation as

$$ed_0 + k - 1 = kp - eMd_1.$$

Let E be the inverse of eM modulo N, e.g. there exists a $c \in \mathbb{N}$ such that $E \cdot eM = 1 + cN$ (if E does not exist, we obtain the factorization of N). Multiplying the above equation by E yields

$$E(ed_0 + k - 1) = (Ek - cqd_1)p - d_1.$$

The only unknown parameter on the left hand side of the equation is k. We make a brute force search for k in the interval $[1, e)$. The correct guess of k gives us a multiple of p up to an additive error $d_1 \leq 2N^{\frac{1}{4}}$. For the correct guess of k, the value $E(ed_0 + k - 1)$ is an approximation of $\bar{k}p$, $\bar{k} = Ek - cqd_1$ up to an error of at most $2N^{\frac{1}{4}}$.

Then an application of Theorem 10 yields the factorization of N. Note that q divides the term $Ek - cqd_1$ iff q divides k which is easily testable (q cannot divide k in the case $e < q$), i.e., the restrictions of Theorem 10 are satisfied.

We briefly summarize our factorization algorithm.

Algorithm LSB-Attack for d_p

INPUT: – (N, e), where $N = pq$ and d_p satisfies $ed_p = 1 \bmod p - 1$

– d_0, M with $d_0 = d_p \bmod M$ and $M \geq N^{\frac{1}{4}}$

1. Compute $E = (eM)^{-1} \bmod N$. If the computation of E fails, output $\gcd(eM, N)$ and $\frac{N}{\gcd(eM,N)}$.
2. FOR $k = 1$ TO e

   Run the algorithm of Theorem 10 on input $E(ed_0 + k - 1)$. If the algorithm's output is p, q then EXIT.

OUTPUT: p, q

The running time of the algorithm is $e \cdot \mathrm{poly}(\log N)$, which concludes the proof. □

In our second approach, we consider the case when MSBs of d_p are known.

Theorem 62 *Let (N, e) be an RSA public key with secret key d and $e = N^{\alpha}$ for some $a \in [0, \frac{1}{4}]$. Further, let $d_p := d \bmod p - 1$. Suppose we are given $\tilde{d}$ with*

$$|d_p - \tilde{d}| \leq N^{\frac{1}{4}-\alpha}.$$

Then N can be factored in polynomial time.

Proof: We start again by looking at the equation $ed_p - 1 = k(p-1)$. Since $d_p < p - 1$, we know that $k < N^{\alpha}$, which implies that q cannot divide k. Compute $\tilde{p} = e\tilde{d} - 1$. Now, $\tilde{p}$ is an approximation of kp up to an additive error of at most

$$|kp - \tilde{p}\,| = |e(d_p - \tilde{d}) + k| \leq N^{\frac{1}{4}} + N^{\alpha} \leq 2N^{\frac{1}{4}}.$$

Thus, $\tilde{p}$ is an approximation of kp with error at most $2N^{\frac{1}{4}}$. Applying the algorithm of Theorem 10 yields the factorization of N.

Here is the algorithm of the attack.

Algorithm MSB-Attack for d_p

INPUT: – (N, e), where $N = pq$ and d_p satisfies $ed_p = 1 \bmod p - 1$

– $\tilde{d}$ with $|d_p - \tilde{d}| \leq N^{\frac{1}{4}-\alpha}$, where $\alpha = \log_N(e)$.

1. Compute $\tilde{p} = e\tilde{d} - 1$.
2. Run the algorithm of Theorem 10 on the input $\tilde{p}$.

OUTPUT: p, q

The algorithm runs in time polynomial in $\log(N)$, which concludes the proof. ◻

6.6 Considering moduli of the form $N = p^r q$

In this section, we consider public key cryptosystems that use moduli of the form $N = p^r q$. Examples for such schemes can be found in [30, 54, 67]. Here, we will mainly focus on RSA-type schemes with the public key equation $ed = 1 \bmod \phi(N)$ and on Takagi's scheme [67]. Notice that in Takagi's scheme only the values $d_p := d \bmod p - 1$ and $d_q := d \bmod q - 1$ are used.

In 1999, Boneh, Durfee and Howgrave-Graham showed that moduli of the form $N = p^r q$ should be handled with care. We have already presented their result in Chapter 3 (see Theorem 13), where we showed that it is a direct application of Coppersmith's theorem for the univariate modular case (Theorem 7). Let us here recall the result of Boneh, Durfee and Howgrave-Graham.

Theorem 13 (BDH) *Let $N = p^r q$, where r is a known constant and p, q are of the same bit-size. Let k be an (unknown) integer that is not a multiple of $p^{r-1}q$. Suppose we know an integer $\tilde{p}$ with*

$$|\, kp - \tilde{p}\,| \leq N^{\frac{r}{(r+1)^2}}.$$

Then N can be factored in polynomial time.

We see that moduli of the form $N = p^r q$ are much more susceptible to attacks that leak bits of p than RSA moduli $N = pq$. In the following sections, our goal is to show that moduli $N = p^r q$ are also more susceptible to attacks that leak bits of the secret key d or of $d_p = d \bmod p - 1$.

6.6.1 Partial key exposure attacks from new attacks for small d

It was stated in the work of Takagi [67] that RSA-type schemes which use the equation $ed = 1 \bmod \phi(N)$ seem to be less vulnerable to Wiener's attack for small decryption exponents d than the original RSA scheme. Namely, Takagi proved a generalized Wiener-like bound of $d \leq N^{\frac{1}{2(r+1)}}$. However, we introduce two attacks that show that there exist much better bounds when small d's are applied. Our first attack is an application of Theorem 13 and yields an improved bound of

$$d \leq N^{\frac{r}{(r+1)^2}} \quad \text{for } r \geq 2.$$

Let us compare the results for $r = 2$: Takagi's method requires that $d \leq N^{\frac{1}{6}}$ whereas our new method works whenever $d \leq N^{\frac{2}{9}}$.

Our second method directly makes use of Coppersmith's method in the univariate case (i.e., Theorem 7) and leads to the bound

$$d \leq N^{\left(\frac{r-1}{r+1}\right)^2} = N^{1-\frac{4r}{(r+1)^2}} \quad \text{for } r \geq 2.$$

Interestingly in contrast to the previous bounds, this new bound converges to N for growing r instead of converging to 1. It improves upon our first attack for all parameter choices $r \geq 3$: The second attack requires that $d \leq N^{\frac{1}{4}}$ in the case $r = 3$ compared to $d \leq N^{\frac{3}{16}}$ for the first method. Thus, our first attack is only superior to the other methods in the case $r = 2$, but moduli of the form $N = p^2 q$ are frequently used in cryptography and therefore they represent one of the most interesting cases.

We want to point out that these new attacks are normally not a threat to Takagi's scheme because there is no need to choose a small decryption exponent d in the scheme. Since in the decryption process only the values d_p and d_q are used, it would be preferable to make these values small instead of the term d. We study partial key exposure attacks on the value d_p in Section 6.6.2.

Interestingly, the new attacks for small decryption exponents d have two features which the original Wiener attack (see Section 4.2) and the Boneh-Durfee attack (see Section 7.2) do not possess:

- One cannot counteract the new attacks by choosing large public exponents e, since the attacks are independent of the value of e. In comparison, Wiener's attack and the Boneh-Durfee attack assume that $e < \phi(N)$. It is known that these attacks cannot be applied if $e > N^{1.5}$ respectively $e > N^{1.875}$.

- The new attacks immediately imply a partial key exposure attack for d with known MSBs. Namely, it makes no difference in the attacks whether the most significant bits of d are zero (and thus d is a small decryption exponent) or are known to the attacker. In contrast, Wiener's attack and the Boneh-Durfee attack for small decryption exponents do not work when the MSBs are non-zero but known. In addition, the new attacks also provide partial key exposure attacks for known LSBs.

  The resulting partial key exposure attacks share the same property as the underlying attacks for small decryption exponents d: They do not rely on the size of the public exponent e. Note that all the partial key exposure attacks mentioned so far were dependent on e and did not work for arbitrary $e \in \mathbb{Z}^*_{\phi(N)}$ (for an overview see Figure 6.4). The new methods are the first partial key exposure attacks that work for all public exponents e.

The difference of the new attacks versus the Wiener and Boneh-Durfee attack is that they do not require to compute the unknown parameter k in the equation $ed - 1 = k\phi(N)$ with Coppersmith's method. Thus, k must not be a small parameter and hence the parameters e and d can be increased (thereby increasing k) without affecting the usability of the attacks.

The reason that these new attacks do not require the direct computation of k is mainly that for moduli $N = p^r q$ the size of the multiplicative group $\mathbb{Z}_N^*$ is $\phi(N) = p^{r-1}(p-1)(q-1)$. Thus for $r \geq 2$, $\phi(N)$ and N share the common divisors p and p^{r-1} and this can be used in the attacks by constructing polynomials with small roots modulo p (our first attack) or modulo p^{r-1} (our second attack), respectively. But looking at the equation $ed - 1 = k\phi(N)$ modulo p or modulo p^{r-1} removes in both cases the unknown parameter k.

The attack modulo p

We present an attack for small decryption exponents d and afterwards extend this approach to partial key exposure attacks. Notice that the following theorem holds for public exponents $e \in \mathbb{Z}$, $\gcd(e, \phi(N)) = 1$ of arbitrary size.

Theorem 63 *Let $N = p^r q$, where $r \geq 2$ is a known constant and p, q are primes of the same bit-size. Let $(e, d) \in \mathbb{Z} \times \mathbb{Z}_N$ be the public-key/secret-key pair satisfying $ed = 1 \bmod \phi(N)$. Suppose that*

$$d \leq N^{\frac{r}{(r+1)^2}}.$$

Then N can be factored in probabilistic polynomial time.

Proof: We know that $\phi(N) = p^{r-1}(p-1)(q-1)$ and therefore the key pair (e, d) satisfies the equation

$$ed - 1 = kp^{r-1}(p-1)(q-1) \quad \text{for some } k \in \mathbb{N}. \tag{6.7}$$

Let E be the inverse of e modulo N, i.e., $Ee = 1 + cN$ for some $c \in \mathbb{N}$. If E does not exist then $\gcd(e, N)$ must be a non-trivial divisor of N.

Note that each possible divisor p^s, $p^s q$ or q $(1 \leq s \leq r)$ does immediately yield the complete factorization of N: p^s can be easily factored by guessing s and taking the s^{th} root over the integers. On the other hand, $p^s q$ yields $\frac{N}{p^s q} = p^{r-s}$ which reduces this case to the previous one. Similarly, q gives us p^r.

Thus, we can assume wlog that the inverse E of e modulo N exists. Multiplying equation (6.7) by E leads to

$$d - E = (Ekp^{r-2}(p-1)(q-1) - cp^{r-1}qd)p.$$

Thus, E is a multiple of p up to an additive error of $d \leq N^{\frac{r}{(r+1)^2}}$. In order to apply Theorem 13, it remains to show that the term $Ekp^{r-2}(p-1)(q-1) - cp^{r-1}qd$ is not a multiple of $p^{r-1}q$. Since $p^{r-1}q$ divides the second term, this is equivalent to show that $Ek(p-1)(q-1)$ is not a multiple of pq. By assumption, we have $\gcd(E, N) = 1$ and thus it remains to prove that pq does not divide $k(p-1)(q-1)$. Assume $k(p-1)(q-1) = c'pq$ for some $c' \in \mathbb{N}$. Then equation (6.7) simplifies to

$$ed - 1 = c'N.$$

On the other hand we know that $eE - 1 = cN$. Combining both equalities we obtain that $d = E \bmod N$. Since $d, E < N$ we have $d = E$ even over $\mathbb{Z}$. But this means that E equals the secret key which yields the factorization of N using Theorem 2.

We briefly summarize the algorithm.

Algorithm (Mod p)-attack for small d using a modulus $N = p^r q$

INPUT: (N, e), where $N = p^r q$ and $ed = 1 \bmod \phi(N)$ for some $d \leq N^{\frac{r}{(r+1)^2}}$.

1. Compute $E = e^{-1} \bmod N$. If the computation of E fails, output p, q.
2. Run the algorithm of Theorem 13 on the input E. If the algorithm's output is p, q then EXIT.
3. Otherwise run the probabilistic algorithm of Theorem 2 on the input (N, e, E).

OUTPUT: p, q

Since every step of the algorithm runs in (probabilistic) polynomial time, this concludes the proof of the theorem. $\square$

Theorem 63 gives us a polynomial time factoring algorithm whenever a certain amount of the MSBs of d are zero. The following corollary shows how the proof of Theorem 63 can be easily generalized such that the result does not only hold if the MSBs of d are zero but instead if they are known to the attacker. This gives as a partial key exposure attack for known MSBs with an analogous bound.

Corollary 64 *(MSBs) Let $N = p^r q$, where $r \geq 2$ is a known constant and p, q are primes of the same bit-size. Let $(e, d) \in \mathbb{Z}\times\mathbb{Z}_N$ be the public-key/secret-key pair satisfying $ed = 1 \bmod \phi(N)$. Suppose we are given $\tilde{d}$ such that*

$$|\, d - \tilde{d} \,| \leq N^{\frac{r}{(r+1)^2}}.$$

Then N can be factored in probabilistic polynomial time.

Proof: The key-pair (e, d) satisfies the equality

$$e(d - \tilde{d}) + e\tilde{d} - 1 = kp^{r-1}(p-1)(q-1) \quad \text{for some } k \in \mathbb{N}.$$

Let $E := e^{-1} \bmod N$, i.e., $Ee = 1 + cN$ for some $c \in \mathbb{N}$. If E does not exist, we obtain the factorization of N. Multiplying the above equation by E yields

$$(d - \tilde{d}) + E(e\tilde{d} - 1) = (Ekp^{r-2}(p-1)(q-1) - cp^{r-1}q(d - \tilde{d}))p.$$

Thus, $E(e\tilde{d} - 1)$ is a multiple of p up to an additive error of $|d - \tilde{d}| \leq N^{\frac{r}{(r+1)^2}}$. The rest of the proof is completely analogous to the proof of Theorem 63. $\Box$

Corollary 64 implies that one has to know roughly a fraction of $1 - \frac{r}{(r+1)^2}$ of the MSBs of d for our partial key exposure attack. We can also derive a partial key exposure attack for known LSBs with an analogous bound.

Corollary 65 *(LSBs) Let $N = p^r q$, where $r \geq 2$ is a known constant and p, q are primes of the same bit-size. Let $(e, d) \in \mathbb{Z} \times \mathbb{Z}_N$ be the public-key/secret-key pair satisfying $ed = 1 \bmod \phi(N)$. Suppose we are given d_0, M with $d = d_0 \bmod M$ and*

$$M \geq N^{1-\frac{r}{(r+1)^2}}.$$

Then N can be factored in probabilistic polynomial time.

Proof: Let us write $d = d_1 M + d_0$, were the unknown d_1 satisfies $d_1 = \frac{d-d_0}{M} < \frac{N}{M} \leq N^{\frac{r}{(r+1)^2}}$. We have the key equation

$$ed_1 M + ed_0 - 1 = kp^{r-1}(p-1)(q-1) \quad \text{for some } k \in \mathbb{N}.$$

Multiply the equation by $E = (eM)^{-1} \bmod N$. We see that $E(ed_0 - 1)$ is a multiple of p up to an additive error of $|d_1| < N^{\frac{r}{(r+1)^2}}$. The rest of the proof is analogous to the proof of Theorem 63. $\square$

Attack modulo p^{r-1}

Our first attack applied Theorem 13 which in turn uses a polynomial with small roots modulo p. In our second attack we will construct a polynomial with a small root modulo p^{r-1} and directly apply Coppersmith's method in the univariate case (Theorem 7). This approach yields better results than the first one whenever $r \geq 3$.

Theorem 66 *Let $N = p^r q$, where $r \geq 2$ is a known constant and p, q are primes of the same bit-size. Let $(e, d) \in \mathbb{Z} \times \mathbb{Z}_N$ be the public-key/secret-key pair satisfying $ed = 1 \bmod \phi(N)$. Suppose that*

$$d \leq N^{\left(\frac{r-1}{r+1}\right)^2}.$$

Then N can be factored in probabilistic polynomial time.

Proof: The key pair (e, d) satisfies the equation

$$ed - 1 = kp^{r-1}(p-1)(q-1) \quad \text{for some } k \in \mathbb{N}.$$

Let E be the inverse of e modulo N, i.e., $Ee = 1 + cN$ for some $c \in N$. In the case that E does not exist, $\gcd(e, N)$ yields the complete factorization of N as shown in the proof of Theorem 63. Multiplying our equation by E leads to

$$d - E = (Ek(p-1)(q-1) - cdpq)p^{r-1}.$$

This gives us a simple univariate polynomial

$$f_{p^{r-1}}(x) = x - E$$

with the root $x_0 = d$ modulo p^{r-1}.

That is, we have a polynomial $f_{p^{r-1}}$ of degree $\delta = 1$ with a root x_0 modulo p^{r-1}. In order to apply Theorem 7, we have to find a lower bound for p^{r-1} in terms of N.

Since p and q are of the same bit-size, we know that $p \geq \frac{1}{2}q$. Hence $p^{r-1} = \frac{N}{pq} \geq \frac{N}{2p^2}$. This gives us

$$p^{r-1} \geq \left(\frac{1}{2}N\right)^{\frac{r-1}{r+1}} \geq \frac{1}{2}N^{\frac{r-1}{r+1}}.$$

Thus, we can choose $\beta = \frac{r-1}{r+1} - \frac{1}{\log N}$ and apply Theorem 7 with the parameter choice β, δ and $c_N = 2$. We can find all roots x_0 that are in absolute value smaller than

$$2N^{\frac{\beta^2}{\delta}} = 2N^{\left(\frac{r-1}{r+1}\right)^2 - \frac{2(r-1)}{(r+1)\log N} + \frac{1}{\log^2 N}} \geq 2N^{\left(\frac{r-1}{r+1}\right)^2 - \frac{1}{\log N}} = N^{\left(\frac{r-1}{r+1}\right)^2}.$$

Hence, we obtain the value $x_0 = d$. Applying the algorithm of Theorem 2 gives us the factorization of N in probabilistic polynomial time.

Remark 67 *Another (deterministic) polynomial time method to find the factorization of N could be the computation of $\gcd(ed-1, N)$. Since $ed - 1 = kp^{r-1}(p-1)(q-1)$, the computation yields a non-trivial divisor of N iff pq does not divide $k(p-1)(q-1)$, which is unlikely to happen. As shown in the proof of Theorem 63, a non-trivial divisor of N reveals the complete factorization of the modulus. So in practice, one might try this alternative gcd-method first and if it fails, one applies the probabilistic algorithm of Theorem 2.*

Let us summarize our new factorization algorithm.

Algorithm (Mod p^r)-attack for small d using a modulus $N = p^r q$

INPUT: (N, e), where $N = p^r q$ and $ed = 1 \bmod \phi(N)$ for some $d \leq N^{\left(\frac{r-1}{r+1}\right)^2}$.

1. Compute $E = e^{-1} \bmod N$. If E does not exist, compute $\gcd(e, N)$ and output p, q.
2. Apply the algorithm of Theorem 7 on input N, $f_{p^{r-1}} = x - E$, $\beta = \frac{r-1}{r+1} - \frac{1}{\log N}$ and $c_N = 2$. This gives us the value d.
3. If the computation $\gcd(ed-1, N)$ yields the factorization, EXIT.
4. Apply the algorithm of Theorem 2 on input (N, e, d).

OUTPUT: p, q

Every step of the algorithm can be computed in polynomial time, which concludes the proof of Theorem 66. $\square$

Similar to the first attack (the (Mod p)-attack) for small decryption exponent d, we can also easily derive partial key exposure attacks for the new attack of Theorem 66. The proof of Theorem 66 shows that in order to find the factorization of N, it suffices to find a linear, univariate polynomial $f_{p^{r-1}}(x) = x + c$ with a root $x_0 \leq N^{\left(\frac{r-1}{r+1}\right)^2}$ modulo p^{r-1}.

We will show that this requirement is easily satisfied for the following partial key exposure attacks. Instead of using small decryption exponents $d < N^{\left(\frac{r-1}{r+1}\right)^2} = N^{1-\frac{4r}{(r+1)^2}}$, the attacker has to know a fraction of roughly $\frac{4r}{(r+1)^2}$ of the bits of N in order to succeed.

Corollary 68 *(MSB) Let $N = p^r q$, where $r \geq 2$ is a known constant and p, q are primes of the same bit-size. Let $(e, d) \in \mathbb{Z} \times \mathbb{Z}_N$ be the public-key/secret-key pair satisfying $ed = 1 \bmod \phi(N)$. Given $\tilde{d}$ with*

$$|\, d - \tilde{d} \,| \leq N^{\left(\frac{r-1}{r+1}\right)^2}.$$

Then N can be factored in probabilistic polynomial time.

Proof: We know that

$$e(d - \tilde{d}) + e\tilde{d} - 1 = 0 \bmod \phi(N),$$

and $\phi(N)$ is a multiple of p^{r-1}. Multiply the equation by $E = e^{-1} \bmod N$, which gives us the desired linear polynomial

$$f_{p^{r-1}}(x) = x + E(e\tilde{d} - 1)$$

with the small root $|x_0| = |d - \tilde{d}| \leq N^{\left(\frac{r-1}{r+1}\right)^2}$ modulo p^{r-1}. The rest of the proof is analogous to the proof of Theorem 66. $\square$

In a similar fashion, we derive a partial key exposure attack for known LSBs.

Corollary 69 *(MSB) Let $N = p^r q$, where $r \geq 2$ is a known constant and p, q are primes of the same bit-size. Let $(e, d) \in \mathbb{Z} \times \mathbb{Z}_N$ be the public-key/secret-key pair satisfying $ed = 1 \bmod \phi(N)$. Given d_0, M with $d = d_0 \bmod M$ and*

$$M \geq N^{\frac{4r}{(r+1)^2}}.$$

Then N can be factored in probabilistic polynomial time.

Proof: Let us write $d = d_1 M + d_0$. Then the unknown parameter satisfies $d_1 < \frac{N}{M} \leq N^{\left(\frac{r-1}{r+1}\right)^2}$. For the key-pair (e, d) we have

$$e(d_1 M + d_0) - 1 = 0 \bmod \phi(N),$$

where $\phi(N)$ is a multiple of p^{r-1}. Multiplying this equation by $E = (eM)^{-1} \bmod N$ gives us the desired linear polynomial

$$f_{p^{r-1}}(x) = x + E(ed_0 - 1)$$

with the small root d_1 modulo p^{r-1}. The rest of the proof is analogous to the proof of Theorem 66. $\square$

6.6.2 Attacks for small d modulo $p-1$

The attacks that we consider in this section for moduli $N = p^r q$ can be considered as a generalization of the attacks that were presented in Section 6.5. Instead of using Theorem 10, we apply the generalized method of Theorem 13 for moduli $N = p^r q$ in our approach.

We derive simple partial key exposure attacks for small public exponents e in both cases: known MSBs and known LSBs. The new attacks are a threat to schemes that use CRT-decoding (for instance Takagi's scheme) in combination with small public exponents.

Let us state our LSB-attack.

Theorem 70 *Let $N = p^r q$, where $r \geq 1$ is a known constant and p, q are primes of the same bit-size. Let e be the public key and let d_p satisfy $ed_p = 1 \bmod p-1$. Suppose we are given d_0, M with $d_0 := d_p \bmod M$ and*

$$M \geq 2N^{\frac{1}{(r+1)^2}}.$$

Then N can be factored in time $e \cdot poly(\log(N))$.

Proof: Let us consider the equation

$$ed_p - 1 = k(p-1) \quad \text{for some } k \in \mathbb{Z}.$$

Since $d_p < (p-1)$, we obtain the inequality $k < e$. Let us write $d_p = d_1 M + d_0$. We can bound the unknown d_1 by $d_1 < \frac{p}{M} \leq N^{\frac{r}{(r+1)^2}}$. Our equation above can be rewritten as

$$ed_1 M + ed_0 + k - 1 = kp.$$

Compute the inverse E of eM modulo N, i.e., $EeM = 1 + cN$ for some $c \in \mathbb{N}$. If E does not exist, we obtain from $\gcd(eM, N)$ the complete factorization of N as shown in Theorem 63. Multiplying our equation with E leaves us with

$$d_1 + E(ed_0 + k - 1) = (Ek - cp^{r-1}qd_1)p.$$

Thus, $E(ed_0 + k - 1)$ is a multiple of p up to some additive error $d_1 \leq N^{\frac{r}{(r+1)^2}}$. Since the parameter k is unknown, we have to do a brute force search for k in the interval $[1, e)$.

In order to apply Theorem 13, it remains to show that the term $(Ek - cp^{r-1}qd_1)$ is not a multiple of $p^{r-1}q$. This is equivalent to the condition that $p^{r-1}q$ does not divide Ek, but we know that $\gcd(E, N) = 1$. Thus, we obtain the condition that $p^{r-1}q$ does not divide k. But $p^{r-1}q$ cannot divide k in the case $e \leq p^{r-1}q$ and otherwise we can easily check the condition by computing $\gcd(k, N)$ for every possible k. The algorithm of Theorem 13 yields the factorization of N for the correct guess of k.

We briefly summarize our factorization algorithm.

Algorithm LSB-Attack for d_p and moduli $N = p^r q$

INPUT: – (N, e), where $N = p^r q$ and d_p satisfies $ed_p = 1 \bmod p-1$

– d_0, M with $d_0 := d_p \bmod M$ and $M \geq 2N^{\frac{1}{(r+1)^2}}$

1. Compute $E = (eM)^{-1} \bmod N$. If the computation of E fails, find the factors p, q of N using $\gcd(eM, N)$.
2. FOR $k = 1$ TO e
   a) If $\gcd(k, N) > 1$ find the factors p, q.
   b) Run the algorithm of Theorem 13 on input $E(ed_0 + k - 1)$. If the algorithm's output is p, q then EXIT.

OUTPUT: p, q

The running time of the algorithm is $e \cdot \mathrm{poly}(\log N)$, which concludes the proof. □

Note that our method from Theorem 70 is polynomial time for public exponents of the size $\mathrm{poly}(\log(N))$ and requires only a $\frac{1}{(r+1)^2}$-fraction of the bits (in terms of the size of N). The following theorem gives us a similar result for partial key exposure attacks with known MSBs, but in contrast the method is polynomial time for all public exponents $e < N^{\frac{r}{(r+1)^2}}$.

Let $e = N^\alpha$. We show that an approximation of d_p up to $N^{\frac{r}{(r+1)^2}-\alpha}$ suffices to find the factorization of N. Note that d_p is of size roughly $N^{\frac{1}{r+1}}$. Hence in the case $\alpha = 0$, a fraction of $\frac{1}{r+1} - \frac{r}{(r+1)^2} = \frac{1}{(r+1)^2}$ of the bits is enough.

Theorem 71 *Let $N = p^r q$, where $r \geq 1$ is a known constant and p, q are primes of the same bit-size. Let $e = N^\alpha$, $\alpha \in [0, \frac{r}{(r+1)^2}]$ be the public key and let d_p satisfy $ed_p = 1 \bmod p-1$. Suppose we are given $\tilde{d}$ with*

$$|\, d_p - \tilde{d}\,| \leq N^{\frac{r}{(r+1)^2}-\alpha}.$$

Then N can be factored in polynomial time.

Proof: We know that

$$ed_p - 1 = k(p-1) \quad \text{for some } k \in \mathbb{N},$$

with $k < e$. The term $e\tilde{d}$ is an approximation of kp up to an additive error of

$$
\begin{aligned}
|kp - e\tilde{d}| &= |e(d_p - \tilde{d}) + k - 1| \leq |e(d_p - \tilde{d})| + |k-1| \\
&\leq N^{\frac{r}{(r+1)^2}} + N^{\alpha} \leq 2N^{\frac{r}{(r+1)^2}}.
\end{aligned}
$$

Thus, one of the terms $e\tilde{d} \pm N^{\frac{r}{(r+1)^2}}$ satisfies the bound of Theorem 13. Note that the algorithm of Theorem 13 can be applied since $k < e < N^{\frac{r}{(r+1)^2}}$ and thus k cannot be a multiple of $p^{r-1}q = \Omega(N^{\frac{r}{r+1}})$.

Let us briefly summarize the factorization algorithm.

MSB-Attack for d_p and moduli $N = p^r q$

INPUT: – (N, e), where $N = p^r q$ and d_p satisfies $ed_p = 1 \bmod p-1$

– $\tilde{d}$ with $|d_p - \tilde{d}| \leq N^{\frac{r}{(r+1)^2} - \alpha}$, where $\alpha = \log_N(e)$.

1. Compute $\tilde{p} = e\tilde{d}$.
2. Run the algorithm of Theorem 13 on the input $\tilde{p} + N^{\frac{r}{(r+1)^2}}$. If the algorithm's output is p, q then EXIT.
3. Otherwise run the algorithm of Theorem 13 on the input $\tilde{p} - N^{\frac{r}{(r+1)^2}}$.

OUTPUT: p, q

The algorithm runs in time polynomial in $\log(N)$, which concludes the proof. $\square$

7 Improved Lattice Bases

"We [he and Halmos] share a philosophy about linear algebra; we think basis-free, we write basis-free, but when the chips are down we close the office door and compute with matrices like fury."
Irving Kaplansky

7.1 Introduction

In the previous chapters, we showed several applications of Coppersmith's method for finding small roots of modular polynomial equations, thereby introducing new RSA vulnerabilities. The methodology that we used by applying Coppersmith's method in the multivariate case had (almost) always the same form:

1. Construct a polynomial f in the variables $x_1, \dots x_k$ which has a "small" root $(x_1^0, \dots, x_k^0)$ modulo some number p.

2. Fix an integer m and construct a set $\{g_1, \dots, g_n\}$ of polynomials depending on f such that each g_i has the small root $(x_1^0, \dots, x_k^0)$ modulo p^m. Furthermore, we choose the polynomials in such a way that the ordering $g_1, \dots, g_n$ implies that each polynomial g_i has the same monomials as the polynomials $g_1, \dots, g_{i-1}$ except for one additional monomial. This implies that the coefficient vectors of $g_1, \dots, g_n$ form a lower triangular matrix.

3. Prove that using the L^3-algorithm, we can find at least k vectors with norm smaller than $\frac{p^m}{\sqrt{n}}$ in the lattice L that is spanned by the coefficient vectors $g_1, \dots, g_n$. Heuristically, the root $(x_1^0, \dots, x_k^0)$ can then be found by resultant computations using the k shortest vectors of the L^3-reduced lattice basis.

One should notice that there is a non-necessary restriction in Step 2 of the approach above: There is no need to require that the coefficient vectors of $g_1, \dots, g_n$ form a triangular lattice basis. In fact, we always used this restriction in the previous chapters in order to keep the determinant computations of L simple. From the determinant in turn, we could derive upper bounds for the k shortest vectors in the L^3-reduced basis of Step 3.

If we skip the restriction that each g_i introduces exactly one additional monomial then we obtain a much larger variety in choosing the polynomials, which in turn should help to improve the bound for the size of the small root $(x_1^0, \ldots, x_k^0)$. One can also view this in the following way: First, we choose the polynomials as before and then we eliminate some polynomials g_i, i.e., we choose only a certain subset of all polynomials in our new approach.

This leads immediately to our main goal of this chapter:

Goal: Identify a subset of vectors that optimizes Coppersmith's method.

Unfortunately, we are not able to give general conditions for optimal subsets of vectors in Coppersmith's method. Theoretically, this seems to be a very difficult problem. However in practice it might not be a problem at all to find good subsets of vectors. The reason for that is that the L^3-algorithm often finds much better approximations of shortest vectors than theoretically predicted. Therefore, we can simply apply the L^3-algorithm and look which subsets of vectors it uses in order to find short vectors. If the triangular lattice basis is not optimal then it is likely that the L^3-algorithm does not use all vectors in order to approximate the shortest lattice vectors. Thus, the problem of finding an optimal lattice basis is the problem of finding the theoretically best bounds when using Coppersmith's method.

In this chapter, we give a case study for finding an optimal lattice basis for the polynomial $f(x,y) = x(N+1+y)-1$ with a small root (x_0, y_0) modulo e. This polynomial is important since it is used in Boneh-Durfee's approach [12] that improves Wiener's attack (see Section 4.2) on low secret exponent RSA. Boneh and Durfee presented two different approaches: The first one finds the factorization of N provided that $d < N^{0.284}$ using a triangular lattice basis B_{BD}. The second one makes use of a suitable subset of vectors from B_{BD}, thereby improving the bound to $N^{0.292}$. They introduce the notion of so-called geometrically progressive matrices in order to analyze the non-triangular lattice basis.

Since the publication of Boneh-Durfee's result, many researchers have tried to improve the bound of $N^{0.292}$. Our results of this chapter strongly indicate that in polynomial time one cannot get beyond this bound using only the polynomial $f(x,y) = x(N+1+y)-1$. This does not imply that there is no polynomial time attack on RSA for larger values on d (see for instance Chapter 4) but for the polynomial $f(x,y)$ the bound seems to be sharp.

We introduce a new method to analyze a large class of subsets of the vectors of the original Boneh-Durfee basis B_{BD}. In fact, the choice of these subsets was motivated by various experiments using the Boneh-Durfee approach. The basis B_{BD} can be naturally divided into blocks. We found that the L^3-algorithm always used a linear combination of the last vectors of these blocks for the shortest vectors in the basis. Moreover, the number of vectors that were used by L^3 in each block always formed a strictly decreasing sequence. This special pattern of vectors — which we call from now on a strictly decreasing sequence/pattern — lead us to the conjecture that these subsets are optimal. We analyze strictly decreasing sequences in this chapter.

Unfortunately, it can be shown that among all strictly decreasing sequences of vectors the optimal bound that can be achieved is $d < N^{0.292}$. Thus, we cannot improve upon the bound of Boneh-Durfee, which was originally analyzed using the notion of geometrically progressive matrices [12]. However, our approach has some advantages when compared to the geometrically progressive bases:

- The property of strictly decreasing sequences of vectors can be formulated in a very simple fashion. It also offers a large variety of different, easily constructible lattice bases that can be used.
- We show that after choosing a certain subset of vectors from the original lattice basis B_{BD}, one can again transform the resulting lattice basis into a triangular form. Therefore, the main advantage of our former approach still holds: Determinant computations in the lattice are easy and we can simply derive upper bounds for the shortest vectors in an L^3-reduced basis.
- We made various experiments with the L^3-algorithm on input B_{BD}. In every experiment the shortest vectors of the reduced lattice bases formed a strictly decreasing sequence of vectors in the blocks. Since one can show that the optimal bound for all strictly decreasing sequences of vectors is $N^{0.292}$, this strongly indicates that one cannot improve upon this bound using the polynomial $f(x, y)$.
- Since the notion of strictly decreasing sequences offers a large variety of choosing suitable subsets of B_{BD}, in practice one should choose a subset with small lattice dimension and reasonable bound for d. We introduce such a pattern and show in experiments that one can get attacks for d's that are close to the theoretical bound using lattice dimensions of at most 72.
- The simple but general formulation of our approach allows us to generalize the method easily to similar polynomials. For instance, one could adopt our approach to the trivariate polynomial $f(x, y, z) = x(N + 1 + y + z) - 1$ which was used by Durfee and Nguyen [28] in order to break an unbalanced RSA-scheme with small secret exponent presented at Asiacrypt 1999 [66].

The chapter is organized as follows: First, we introduce the Boneh-Durfee approach and describe the lattice basis B_{BD}, which leads to the bound $N^{0.284}$. Afterwards, we describe our algorithm that chooses strictly decreasing sequences of vectors from B_{BD}. Moreover, we suggest to modify the resulting lattice basis in such a way that it becomes again triangular by eliminating columns in the basis. We prove that this modification is correct: Whenever we have a short vector in the new triangular lattice, then the corresponding short vector in the non-triangular lattice basis — that is constructed using the same coefficients in the linear combination of basis vectors — is also a short vector. In other words: Eliminating some coordinates in the lattice spanned by the non-triangular basis does not change the norm of the lattice vectors significantly.

Furthermore, we analyze the subset of vectors that was chosen by Boneh-Durfee in order to prove the bound $N^{0.292}$ with our new method. It turns out that this subset can easily by analyzed. We also propose another strictly decreasing sequence of vectors that yields lattice bases of small dimension. We demonstrate the usefulness of this new sequence by providing some experimental results.

We conclude the chapter by studying a special subset of B_{BD}, where only x-shifted polynomials of the form $x^i e^{m-k} f^k(x,y)$ are used. Boneh and Durfee noticed that this special subset reproduces the Wiener bound of $d < N^{\frac{1}{4}}$. However, we found out experimentally that Coppersmith's resultant heuristic for finding the root (x_0, y_0) does not work in this case, since the resultants were always zero in our experiments. We are able to almost completely analyze this behavior. Moreover, we give an alternative provable method how to recover (x_0, y_0) and the factorization of N in this case. This can be understood as a warning that the resultant heuristic may fail sometimes, but at the same time it is a motivation to search for provable methods. In our case the resultant heuristic fails because the polynomials corresponding to shortest vectors have a special structure: They are all divisible by a polynomial whose coefficients yield the factorization of N, which is similar to the case of the bivariate, rigorous methods presented in Sections 5.2 and 5.3. Hence, a failure of the resultant heuristic must not automatically imply a failure of a Coppersmith-like attack.

7.2 The Boneh-Durfee lattice

In this section we review the lattice attack by Boneh and Durfee [12] on low exponent RSA. Let $d < N^{\delta}$. We assume that the size of $e \in \mathbb{Z}^*_{\phi(N)}$ is in the order of the size of N. If e is smaller, the attack of Boneh and Durfee becomes even more effective (see [12], Section 5).

Boneh and Durfee start with the RSA key equation

$$ed + k(N + 1 - (p + q)) = 1 \tag{7.1}$$

where $|k| = \frac{ed-1}{\phi(N)} < d$. Let $A = N + 1$. Looking at equation (7.1) modulo e gives us the polynomial

$$f(x, y) = x(A + y) - 1,$$

with the root $(x_0, y_0) = (k, -(p + q))$ modulo e. We define the upper bounds $X := N^{\delta}$ and $Y := 3N^{\frac{1}{2}}$ satisfying $|x_0| \le X$ and $|y_0| \le Y$.

Next, we fix some integer m and define the polynomials

$$g_{i,k}(x,y) := x^i f^k(x,y) e^{m-k} \quad \text{and} \quad h_{j,k}(x,y) := y^j f^k(x,y) e^{m-k}.$$

In the sequel, the polynomials $g_{i,k}$ are referred to as x-shifts and analogously the polynomials $h_{j,k}$ are referred to as y-shifts. By construction, the point (x_0, y_0) is a root of all these polynomials modulo e^m. Thus, we can apply Howgrave's theorem and search for a small norm integer linear combination of polynomials $g_{i,k}(xX, yY)$ and $h_{j,k}(xX, yY)$. This is done by using the L^3-lattice reduction algorithm. The goal is to construct a lattice L that is guaranteed to contain a vector shorter than $e^m/\sqrt{\dim(L)}$.

The Boneh-Durfee lattice is spanned by the coefficient vectors of the polynomials $g_{i,k}, h_{j,k}$ for certain parameters i, j and k. For each $k = 0, \ldots, m$, Boneh and Durfee use the x-shifts $g_{i,k}(xX, yY)$ for $i = 0, \ldots, m-k$. Additionally, they use the y-shifts $h_{j,k}$ for $j = 0, \ldots, t$ for some parameter t.

We call the lattice constructed by Boneh and Durfee the lattice L_{BD}. The basis for L_{BD} is denoted by B_{BD}. The lattice L_{BD} is spanned by the row vectors of B_{BD}. Since the lattice depends on the parameters m and t, we sometimes refer to the basis by $B_{BD}(m,t)$ to clarify notation. It is easy to see that the basis vectors of the lattice L_{BD} form a triangular matrix. We give an example of the lattice basis for the parameter choice $m = 2$ and $t = 1$, e.g. $B_{BD}(2,1)$ is the matrix:

	1	x	xy	x^2	x^2y	x^2y^2	y	xy^2	x^2y^3
e^2	e^2								
xe^2		e^2X							
fe	$-e$	eAX	eXY						
x^2e^2				e^2X^2					
xfe		$-eX$		eAX^2	eX^2Y				
f^2	1	$-2AX$	$-2XY$	A^2X^2	$2AX^2Y$	X^2Y^2			
ye^2							e^2Y		
yfe			$eAXY$				$-eY$	eXY^2	
yf^2			$-2AXY$		A^2X^2Y	$2AX^2Y^2$	Y	$-2XY^2$	X^2Y^3

Boneh and Durfee showed that for $\delta < 0.284$, one can find m, t such that an L^3-reduced basis of L_{BD} contains vectors short enough to apply Howgrave's theorem and factor the modulus N. This was improved in the same work to $\delta < 0.292$ by using non-triangular lattice bases. This is up to now the best bound for the cryptanalysis of low secret exponent RSA. The attack works under the Coppersmith's resultant heuristic: Two polynomials obtained from two different sufficiently short vectors in the reduced basis have a non-vanishing resultant. Although heuristic, no failure of the method for sufficiently large δ is known.

Boneh and Durfee also argue that using $t = 0$, i.e., only x-shifts are used to construct the lattice basis, one obtains already an attack working for $\delta < 0.25$. This reproduces

Wiener's result. However, experiments show that the method of Boneh and Durfee never works when using only x-shifts. In Section 7.5, we will explain why this is the case. Of course, this failure of the Boneh-Durfee method in the special case where only x-shifts are used does not affect the method in general. It only points out that one has to be careful when using Coppersmith's resultant heuristic in the multivariate case. On the other hand, we present another provable method in Section 7.5 that finds the factorization of N in the special case where only x-shifts are used, thereby avoiding the resultant heuristic.

Notations for the lattice basis B_{BD}

Since the lattice L_{BD} is the starting point of our further constructions, we introduce some notations on the rows and columns of the lattice basis B_{BD}.

We refer to the coefficient vectors of the polynomials $g_{i,k}(xX, yY)$ as the X-block. The X-block is further divided into $X_l, l = 0, \dots, m$, blocks, where the block X_l consists of the $l+1$ coefficient vectors of $g_{i,k}$ with $i + k = l$. These $l+1$ vectors are called $X_{l,k}$, that is the k^{th} vectors in the X_l block is the coefficient vector of $g_{l-k,k}$.

The coefficient vectors of the polynomials $h_{j,k}$ form the Y-block. We define the Y_j block as the block of all $m+1$ coefficient vectors of polynomials that are shifted by y^j. The k^{th} vector in the block Y_j is called $Y_{j,k}$, it is identical to the coefficient vector of $h_{j,k}$.

Every column in the basis B_{BD} is labelled by a monomial $x^i y^j$. All column vectors with label $x^l y^j$, $l \geq j$, form the $X^{(l)}$ column block. Analogously, we define the $Y^{(l)}$ column block to consist of all column vectors labelled with $x^i y^{i+l}$.

In the example in Section 7.2, the horizontal lines divide the basis $B_{BD}(2, 1)$ into the blocks X_1, X_2, X_3 and Y_1. Analogously, the vertical lines divide $B_{BD}(2, 1)$ into the column blocks $X^{(1)}$, $X^{(2)}$, $X^{(3)}$ and $Y^{(1)}$. In this example, the basis entry in row $Y_{1,2}$ and column x^2y is A^2X^2Y.

7.3 A new method for selecting basis vectors

We introduce an alternative method for modifying the lattice bases B_{BD}. This new method yields the same bound $\delta < 0.292$ as the Boneh-Durfee approach using geometrically progressive matrices. However, it has several advantages compared to their method.

1. Our method is more flexible than the Boneh-Durfee approach. It can be used to analyze the Boneh-Durfee lattice bases as well as other bases. For instance for suitable parameter choices, our method can be used to analyze lattice bases with significantly reduced lattice dimension as a function of m and t. The practical implication is that we are able to get closer to the theoretical bound. We give

experimental results for $\delta > 0.265$, which was the largest bound given in the experiments of Boneh-Durfee.

2. Our proofs are elementary. Furthermore, as opposed to the Boneh-Durfee lattice for $\delta < 0.292$, the lattice bases we use in the attacks remain triangular. Hence, our determinant computations are simple.

3. Third, our construction makes use of structural properties of the underlying polynomials. Thus, it should also apply to other lattice constructions that use similar polynomials.

Our method transforms the Boneh-Durfee lattice basis $B_{BD}(m,t)$ into a new basis $B(m,t)$ for a lattice L of smaller dimension.

Algorithm Lattice Basis Construction (LBC)

INPUT: Lattice basis $B_{BD}(m,t)$

1. Fix a strictly decreasing pattern $(p_0, p_1, \dots, p_t)$, $p_0 > p_1 > \dots > p_t$.

2. In the Y_t block of the basis B_{BD} remove every vector except for the last p_t vectors $Y_{t,m-p_t+1}, \dots, Y_{t,m}$. In the Y_{t-1} block remove every vector except for the last p_{t-1} vectors $Y_{t,m-p_{t-1}+1}, \dots, Y_{t,m}$, and so on. Finally, in the Y_1 block remove every vector except for the last p_1 vectors $Y_{m-p_1+1}, \dots, Y_m$.

3. Remove every vector in the X-block except for the vectors in the p_0 blocks X_{m-p_0+1}, X_{m-p_0+2}, $\dots, X_m$.

4. Delete columns in such a way that the resulting basis is again triangular. That means: Remove all column blocks $X^{(0)}, X^{(1)}, \dots, X^{(m-p_0)}$. Furthermore, in the column block $Y^{(l)}$, $l = 1, \dots, t$, remove the columns labelled with $x^i y^{i+l}$ for $0 \le i < m - p_t$.

OUTPUT: Lattice basis $B(m,t)$

This construction leads to a triangular basis $B(m,t)$ of a new lattice L, which will be used in our approach. As a short-hand notation for $B(m,t)$ we usually write B.

As opposed to Boneh and Durfee, we do not integrate more y-shifts to improve the bound $\delta < 0.284$, instead we remove some x-shifts.

Notice that by choosing the strictly decreasing pattern $(p_0, p_1, \dots, p_t) = (t+1, t, \dots, 1)$, we get a bases $B(m,t)$ with minimal dimension. Applying the construction with this special pattern to the example given in Section 7.2, we obtain the following lattice basis of L with parameters $m = 2$ and $t = 1$.

$$B(2,1) = \left[\begin{array}{c|cc|ccc|c}
 & x & xy & x^2 & x^2y & x^2y^2 & x^2y^3 \\
\hline
xe^2 & e^2X & & & & & \\
fe & eAX & eXY & & & & \\
\hline
x^2e^2 & & & e^2X^2 & & & \\
xfe & -eX & & eAX^2 & eX^2Y & & \\
f^2 & -2AX & -2XY & A^2X^2 & 2AX^2Y & X^2Y^2 & \\
\hline\hline
yf^2 & & -2AXY & & A^2X^2Y & 2AX^2Y^2 & X^2Y^3
\end{array}\right]$$

Let $\bar{B}$ be the non-triangular basis we obtain after Step 3 of the construction in Algorithm LBC. That is, $\bar{B}$ consists of the remaining basis vectors of B_{BD} after removing row vectors but without removing columns. The lattice spanned by the row vectors of $\bar{B}$ is called $L_{\bar{B}}$. We adopt the notations of Section 7.2 for the rows and columns of B and $\bar{B}$. For example, the row vector $X_{l,k}$ of B is the coefficient vector of $g_{l-k,k}$, where we removed all the entries specified in Step 4 of the construction. In the basis $B(2,1)$ above, the row vector $X_{2,0}$ is the vector $(0, 0, e^2X^2, 0, 0, 0)$.

We call a column vector x^iy^j that appears in the basis $\bar{B}$ but not in the basis B a *removed column* of B. The bases B and $\bar{B}$ are constructed using the same coefficient vectors, where in B certain columns are removed. Having a vector $u = \sum_{b\in B} c_b b$ in the span of B, one can compute the corresponding linear combination $\bar{u} = \sum_{b\in\bar{B}} c_b b$ of vectors in $\bar{B}$ with the same coefficients c_b. Hence, the vector dimension of $\bar{u}$ is larger than the vector dimension of u. One can regard the additional vector entries in $\bar{u}$ as a reconstruction of the vector entries of u in the removed columns. Therefore, we call $\bar{u}$ the *reconstruction vector* of u.

The row vectors $X_{l,k}$ ($l = m - p_0 + 1, \ldots, m; k \leq l$) and $Y_{j,k}$ ($j = 1, \ldots, t; k = m - p_j + 1, \ldots, m$) form the basis B. These vectors are no longer the coefficient vectors of the polynomials $g_{l-k,k}(xX, yY)$ and $h_{j,k}(xX, yY)$, respectively, since we remove columns in Step 4 of the construction. However, in order to apply Howgrave's theorem we must ensure that we construct a linear combination of bivariate polynomials that evaluates to zero modulo e^m at the point $(x_0, y_0) = (k, s)$. Hence, we still have to associate the rows $X_{l,k}$ and $Y_{j,k}$ with the polynomials $g_{l-k,k}$ and $h_{j,k}$. The basis vectors of $\bar{B}$ represent the coefficient vectors of these polynomials. Therefore, after finding a small vector $u = \sum_{b\in B} c_b b$ in L, we compute the reconstruction vector $\bar{u} = \sum_{b\in\bar{B}} c_b b$ in $L_{\bar{B}}$. That is, we reconstruct the entries in the removed columns. Once the reconstruction vectors of two sufficiently short vectors in L are computed, the rest of our method equals the Boneh-Durfee method.

In the remainder of this section we show that short vectors u in L lead to short reconstruction vectors $\bar{u}$ in $L_{\bar{B}}$.

Theorem 72 *Let $u := \sum_{b\in B} c_b b$ with $\|u\| < e^m$ be a vector in L. Then the reconstruction vector $\bar{u} = \sum_{b\in \bar{B}} c_b b$ satisfies $\|\bar{u}\| < e^m + \mathcal{O}(\frac{e^m}{XY})$.*

Proof. In order to prove Theorem 72, we first show that removed columns of B are small linear combinations of column vectors in B. We give an example for the removed column x^0y^0 in $B(2,1)$. Applying the construction in the following proof, we see that this column is a linear combination of the columns x^1y^1 and x^2y^2 in B.

$$\begin{pmatrix} 0 \\ -e \\ \hline 0 \\ 0 \\ 1 \\ \hline 0 \end{pmatrix} = -\frac{1}{XY}\begin{pmatrix} 0 \\ eXY \\ \hline 0 \\ 0 \\ -2XY \\ \hline -2AXY \end{pmatrix} - \frac{1}{X^2Y^2}\begin{pmatrix} 0 \\ 0 \\ \hline 0 \\ 0 \\ X^2Y^2 \\ \hline 2AX^2Y^2 \end{pmatrix}$$

Roadmap of the proof of Theorem 72

- We prove that all removed columns in the X^i blocks are linear combinations of columns in B with small coefficients (Lemma 73).
- We show that also all removed columns in the Y^j blocks are linear combinations of columns in B with small coefficients (Lemma 75).
- We express removed columns as linear combinations of columns in B with small coefficients. This implies the following: Whenever a linear combination of rows of B makes every vector entry small (i.e., every integer linear combination of the column vectors is small), then the corresponding linear combination of the removed column must be small as well (Lemma 76 and Lemma 77).

Lemma 73 *All removed columns in the column blocks $X^i, i \le m - p_0$ are linear combinations of columns in B. Moreover, in these linear combinations, the coefficient for a column vector in $X^{(l)}, l > m - p_0$, can be bounded by $\frac{1}{(XY)^{l-i}} \cdot c$, where c depends only on m and t.*

Proof: If x^iy^j is a removed column of B, we show that x^iy^j is a linear combination of columns $x^{i+1}y^{j+1}$, …, x^my^{m-i+j}. If $x^{i+1}y^{i+1}$ is a removed column, we can repeat the argument to show that $x^{i+1}y^{i+1}$ is a linear combination of the remaining columns $x^{i+2}y^{j+2}$, …, x^my^{m-i+j} . Continuing in this way until all removed columns have been represented as linear combinations of columns in B, proves the lemma. Hence, it suffices to prove the following claim.

Claim 1 *If x^iy^j is a removed column of B, then x^iy^j is a linear combination of the columns $x^{i+1}y^{j+1}$, $x^{i+2}y^{j+2}$,..., x^my^{m-i+j}, where the coefficient of column $x^{i+b}y^{j+b}, b = 1, \dots, m-i$, is given by*

$$-\frac{1}{(XY)^b}\binom{j+b}{j}.$$

Note, that the coefficient $c_b = \binom{j+b}{j}$ depends only on m and t, since j depends on m and t.

We will prove Claim 1 by showing that for each row in $B(m,t)$ the entry of the column x^iy^j in this row is a linear combination of the entries of the columns $x^{i+b}y^{j+b}$ in this row, with the coefficients as in the claim. We prove this for the rows in the X-block and Y-block separately.

Let $X_{l,k}$ be a row in block X_l, where $l > m - p_0$. The coefficients in this row are the coefficients of the polynomial $e^{m-k}(xX)^{l-k}f^k(xX, yY)$. By definition of f this polynomial is

$$e^{m-k}(xX)^{l-k}f^k(xX,yY) = e^{m-k}\sum_{p=0}^{k}\sum_{q=0}^{p}(-1)^{k+p}\binom{k}{p}\binom{p}{q}A^{p-q}X^{p+l-k}Y^qx^{p+l-k}y^q. \tag{7.2}$$

To obtain the coefficient of $x^{i+b}y^{j+b}$ in $e^{m-k}(xX)^{l-k}f^k(xX,yY)$, we set $p := i-l+k+b$ and $q := j+b$. Hence, this coefficient is given by

$$\begin{aligned} & e^{m-k}(-1)^{i-l+b}\binom{k}{i-l+k+b}\binom{i-l+k+b}{j+b}A^{i-l+k-j}X^{i+b}Y^{j+b} \\ = \;& e^{m-k}A^{i-l+k-j}X^iY^j(-1)^{i-l}(-1)^b\binom{k}{i-l+k+b}\binom{i-l+k+b}{j+b}(XY)^b. \end{aligned}$$

We can ignore the factor $e^{m-k}A^{i-l+k-j}X^iY^j(-1)^{i-l}$, common to all entries in row $X_{l,k}$ in the columns $x^{i+b}y^{j+b}$. Then Claim 1 restricted to row $X_{l,k}$ reads as

$$\binom{k}{i-l+k}\binom{i-l+k}{j} = \sum_{b=1}^{m-i}(-1)^{b+1}\frac{1}{(XY)^b}\binom{j+b}{j}\binom{k}{i-l+k+b}\binom{i-l+k+b}{j+b}(XY)^b$$

Since the binomial coefficient $\binom{k}{i-l+k+b}$ is non-zero only for $k \geq i-l+k+b$, we only have to sum up to $b \leq l-i$. Substituting $i-l+k$ by i' yields

$$\binom{k}{i'}\binom{i'}{j} = \sum_{b=1}^{k-i'}(-1)^{b+1}\binom{j+b}{j}\binom{k}{i'+b}\binom{i'+b}{j+b}. \tag{7.3}$$

Subtracting the left-hand side, Claim 1 restricted to row $X_{l,k}$ reduces to

$$0 = \sum_{b=0}^{k-i'}(-1)^{b+1}\binom{j+b}{j}\binom{k}{i'+b}\binom{i'+b}{j+b}. \tag{7.4}$$

One checks that
$$\binom{j+b}{j}\binom{k}{i'+b}\binom{i'+b}{j+b} = \frac{k!}{(k-i')!j!(i'-j)!}\binom{k-i'}{b}.$$
This shows
$$\sum_{b=0}^{k-i'}(-1)^{b+1}\binom{j+b}{j}\binom{k}{i'+b}\binom{i'+b}{j+b} = -\frac{k!}{(k-i')!j!(i'-j)!}\sum_{b=0}^{k-i'}(-1)^b\binom{k-i'}{b}.$$
Since $\sum(-1)^b\binom{k-i'}{b} = (1+(-1))^{k-i'} = 0$ we get equation (7.4). This proves Claim 1 for the X-block.

In the same manner, Claim 1 is proved for the Y-block. Let $Y_{l,k}$ be a row in block Y_l. Analogously to equation (7.2), we get
$$e^{m-k}(yY)^l f^k(xX,yY) = e^{m-k}\sum_{p=0}^{k}\sum_{q=0}^{p}(-1)^{k+p}\binom{k}{p}\binom{p}{q}A^{p-q}X^pY^{q+l}x^py^{q+l}.$$
We obtain the coefficients of $x^{i+b}y^{j+b}$ in $e^{m-k}y^l f^k(xX,yY)$ by setting $p := i+b$ and $q := j-l+b$. Again, we ignore the common factors in e, A, X and Y. Now, Claim 1 for $Y_{l,k}$ reduces to
$$\binom{k}{i}\binom{i}{j-l} = \sum_{b=1}^{m-i}(-1)^{b+1}\binom{j+b}{j}\binom{k}{i+b}\binom{i+b}{j-l+b} \qquad (7.5)$$
Notice that both sides of the equation are zero if $k < i$. However, there is a problem for the case $k = i$ because the left-hand side is $\binom{i}{j-l}$ and the right-hand side is zero due to the term $\binom{k}{i+b}$. Hence, the equality only holds for $j < l$. But further notice that the case $k = i$ cannot occur for the basis B: In our lattice basis construction algorithm we use a strictly decreasing pattern $p_0 > p_1 > \cdots > p_t$. But $p_0 > p_i, i > 0$ means that if we remove $m - p_0 + 1$ column blocks X^i, then we remove in each Y_l block, $1 \le l \le t$, at least the first $m - p_0 + 2$ rows $Y_{l,k}$ which implies that $k > i$.

We show in the following that equation (7.5) holds for the parameter settings of our lattice basis B. We only have to sum up to $b \le k - i$, because the factor $\binom{k}{i+b}$ is zero for $k < i + b$. Substituting $j - l$ by j and subtracting the left-hand side yields
$$\begin{aligned} 0 &= \sum_{b=0}^{k-i}(-1)^{b+1}\binom{j+l+b}{j+l}\binom{k}{i+b}\binom{i+b}{j+b} \\ &= \frac{k!l!}{(k-i)!(j+l)!(i-j)!}\sum_{b=0}^{k-i}(-1)^{b+1}\binom{j+l+b}{l}\binom{k-i}{b}. \end{aligned}$$
In order to show that the right-hand side is zero, we prove the following more general lemma.

Lemma 74 *For all $l, m, n \in \mathbb{N}$, we have*

$$\sum_{b=0}^{n}(-1)^b\binom{n}{b}\binom{m+b}{l} = (-1)^n\binom{m}{l-n}.$$

Proof: We prove the lemma by induction over n. For $n = 0$ the statement is correct.

Now let us assume that the statement is correct for $n-1$. We conclude in the inductive step that it also holds for n. Let us write

$$\begin{aligned}\sum_{b=0}^{n}(-1)^b\binom{n}{b}\binom{m+b}{l} &= \sum_{b=0}^{n}(-1)^b\left(\binom{n-1}{b}+\binom{n-1}{b-1}\right)\binom{m+b}{l} \\ &= \sum_{b=0}^{n}(-1)^b\binom{n-1}{b}\binom{m+b}{l}+\sum_{b=0}^{n}(-1)^b\binom{n-1}{b-1}\binom{m+b}{l}\end{aligned}$$

Since $\binom{n-1}{n} = \binom{n-1}{-1} = 0$, we eliminate one term in each summation. Thus, we can rewrite our summations as

$$\sum_{b=0}^{n-1}(-1)^b\binom{n-1}{b}\binom{m+b}{l}+\sum_{b=0}^{n-1}(-1)^{b+1}\binom{n-1}{b}\binom{m+b+1}{l}$$

Using the induction assumption for $n-1$, we obtain

$$(-1)^{n-1}\left(\binom{m}{l-n+1}-\binom{m+1}{l-n+1}\right) = (-1)^n\binom{m}{l-n}.$$

This concludes the proof of Lemma 74. □

Now let us apply Lemma 74 to the sum

$$\sum_{b=0}^{k-i}(-1)^{b+1}\binom{k-i}{b}\binom{j+l+b}{l} = (-1)^{k-i-1}\binom{j+l}{l-k+i}.$$

We observe that this term is zero whenever $k > l+i$. As we argued before (see the paragraph following equation (7.5)) the strictly increasing pattern $p_0 > p_1$ of our lattice basis construction algorithm implies that $k > i$. Now, $p_1 > p_2 > \cdots > p_t$ implies that by going from the Y_l to the Y_{l+1} block we remove at least one more vector, i.e., if $Y_{l,k}$ is the vector in Y_l with minimal k then $Y_{l+1,k}$ cannot be a vector in B. In other words if l increases by one then k increases by at least one which implies the necessary condition $k > l+i$.

This concludes the proof of Lemma 73. □

Lemma 75 *Every removed column vector $x^i y^{i+j}, i \le m - p_j$, is a linear combination of the columns in the column block $Y^{(j)}$ of B. In this linear combination, the coefficient for a column vector $x^k y^{k+j}, k > m - p_j$, can be bounded by $\frac{1}{(XY)^{k-i}} \cdot c$, where c depends only on m.*

Proof: The proof is analogous to the proof of Lemma 73. With the same reasoning as in Lemma 73, it suffices to prove the following claim.

Claim 2 *If $x^i y^{i+j}$ is a removed column of B, then $x^i y^{i+j}$ is a linear combination of the columns $x^{i+1}y^{i+j+1}$, $x^{i+2}y^{i+j+2}, \ldots,$ $x^m y^{m+j}$, where the coefficient of column $x^{i+b}y^{i+j+b}, b = 1, \ldots, m-i$, is given by*

$$-\frac{1}{(XY)^b}\binom{i+b}{i}.$$

Note, that the coefficient $-\binom{i+b}{i}$ depends only on m since $i \le m$.

We will prove Claim 2 by showing that for each row in $B(m,t)$ the entry of the column $x^i y^{i+j}$ in this row is a linear combination of the entries of the columns $x^{i+b}y^{i+j+b}$ in this row, with the coefficients as in the claim.

Let $Y_{l,k}$ be a row in block Y_j, where $j \le l \le t$ and $k > m - p_l$. The coefficients in this row are the coefficients of the polynomial $e^{m-k}(yY)^l f^k(xX, yY)$. By definition of f this polynomial is

$$e^{m-k}(yY)^l f^k(xX, yY) = e^{m-k}\sum_{p=0}^{k}\sum_{q=0}^{p}(-1)^{k+p}\binom{k}{p}\binom{p}{q}A^{p-q}X^pY^{q+l}x^py^{q+l}. \tag{7.6}$$

In order to obtain the coefficient of $x^{i+b}y^{i+j+b}$, we set $p := i+b$ and $q := i+b+j-l$. As in the proof of Lemma 73, we ignore the common factors in e, A, X and Y to simplify our equation. Then Claim 2 restricted to row $Y_{h,k}$ reads as

$$\binom{k}{i}\binom{k}{i+j-l} \;=\; \sum_{b=1}^{m-i}(-1)^{b+1}\frac{1}{(XY)^b}\binom{i+b}{i}\binom{k}{i+b}\binom{i+b}{i+b+j-l}(XY)^b$$

Since $\binom{k}{i+b} = 0$ for $k < i+b$, we only have to sum up the terms with $b \le k-i$. Subtracting the left-hand side of the equation leaves us with

$$\begin{aligned} 0 &= \sum_{b=0}^{k-i}(-1)^{b+1}\binom{i+b}{i}\binom{k}{i+b}\binom{i+b}{i+b+j-l} \\ &= \binom{k}{i}\sum_{b=0}^{k-i}(-1)^{b+1}\binom{k-i}{b}\binom{i+b}{l-j} \end{aligned}$$

Applying Lemma 74 to the sum in the equation above yields

$$\sum_{b=0}^{k-i}(-1)^{b+1}\binom{k-i}{b}\binom{i+b}{h-l} = (-1)^{k-i+1}\binom{i}{l-j-k+i}.$$

It holds that $\binom{i}{l-j-k+i} = 0$ for $k > l+i-j$. In the proof of Lemma 73, we have already seen that every strictly decreasing pattern $p_0 > p_1 > \cdots > p_t$ implies that $k > l+i$. Therefore $k > l+i-j$ which concludes the proof of Lemma 75. $\Box$

Lemma 76 *Let $u := \sum_{b\in B} c_b b$ be a linear combination of vectors in B with $\|u\| < e^m$. For fixed m and t and for every removed column $x^i y^j$ in the $X^{(i)}$ block ($0 \le i \le m-p_0$), the entry $x^i y^j$ in the reconstruction vector $\bar{u} = \sum_{b\in\bar{B}} c_b b$ can be bounded by $\mathcal{O}\left(\frac{e^m}{(XY)^{m-p_0+1-i}}\right)$.*

Proof: Consider a removed column $x^i y^j$. Let $v := (v_1, v_2, \ldots, v_n)^T$ be the column vector $x^i y^j$ in $\bar{B}$, where the entries are multiplied by the coefficients c_b. We want to show that $|\sum_{k=1}^n v_k| = \mathcal{O}(\frac{e^m}{(XY)^{m-t-i}})$. This would prove the lemma.

Apply Lemma 73 and write v as a linear combination of the p_0 columns

$$x^{m-p_0+1}y^{m-p_0+1-i+j}, \ldots, x^m y^{m-i+j}$$

in B, where again the entries in each of the p_0 vectors are multiplied by the coefficients c_b. Call these columns $w_i = (w_{i,1}, \ldots, w_{i,n})^T$ for $i = 1, \ldots, p_0$. Applying Lemma 73 yields

$$v = \frac{d_1}{(XY)^{m-p_0+1-i}}w_1 + \frac{d_2}{(XY)^{m-p_0+2-i}}w_2 + \cdots + \frac{d_{p_0}}{(XY)^{m-i}}w_{p_0}$$

According to Lemma 73, the d_i are constant for fixed m and t. By assumption $\|u\| < e^m$. Hence, all components of u are less than e^m. From this, we obtain $|\sum_k w_{i,k}| < e^m$. This implies

$$\begin{aligned}
\left|\sum_k v_k\right| &= \left|\frac{d_1}{(XY)^{m-p_0+1-i}}\sum_k w_{1,k} + \cdots + \frac{d_{p_0}}{(XY)^{m-i}}\sum_k w_{p_0,k}\right| \\
&\le \left|\frac{d_1}{(XY)^{m-p_0+1-i}}\sum_k w_{1,k}\right| + \cdots + \left|\frac{d_{p_0}}{(XY)^{m-i}}\sum_k w_{p_0,k}\right| \\
&\le \left|\frac{d_1 e^m}{(XY)^{m-p_0+1-i}}\right| + \cdots + \left|\frac{d_{p_0} e^m}{(XY)^{m-i}}\right| \\
&= \mathcal{O}\left(\frac{e^m}{(XY)^{m-p_0+1-i}}\right) + \cdots + \mathcal{O}\left(\frac{e^m}{(XY)^{m-i}}\right)
\end{aligned}$$

Therefore, $|\sum_k v_k|$ can be bounded by $\mathcal{O}\left(\frac{e^m}{(XY)^{m-p_0+1-i}}\right)$. ◻

Lemma 77 *Let $u := \sum_{b\in B} c_b b$ be a linear combination of vectors in B with $\|u\| < e^m$. For fixed m and t and for every removed column $x^i y^{i+j}$ in the $Y^{(j)}$ block $(1 \leq j \leq t, 0 \leq i \leq m - p_j)$, the entry $x^i y^{i+j}$ in the reconstruction vector $\bar{u} = \sum_{b\in \bar{B}} c_b b$ can be bounded by $\mathcal{O}\left(\frac{e^m}{(XY)^{m-p_j+1-i}}\right)$.*

Proof: The proof is completely analogous to the proof of Lemma 76. We apply Lemma 75 and write the removed column $x^i y^{i+j}$ as a linear combination of the p_j columns $x^{m-p_j+1}y^{m-p_j+1+j}, \ldots, x^m y^{m+j}$ in B, where the entries in each of the p_j vectors are multiplied by the coefficients c_b. The remainder of the proof follows the reasoning in Lemma 76 and is therefore omitted. ◻

From Lemmas 76 and 77, we can conclude that if we use the reconstruction vector $\bar{u}$ instead of the short vector u, we do not enlarge the norm significantly: Every entry of a removed column contributes to the norm of $\bar{u}$ with a term of at most $\mathcal{O}(\frac{e^m}{XY})$. This shows the correctness of our approach and concludes the proof of Theorem 72. ◇

7.4 Application of the method

The new lattice bases construction (LBC) method that we proposed in the previous section offers a large variety of different bases that can be used in order to attack RSA with small secret exponent d. There are three key observations how to optimize the choice of a lattice bases:

1. If we analyze the Boneh-Durfee basis B_{BD}, we see that the entries on the diagonal of the X-blocks of B_{BD} imply a Wiener-like bound of $d \leq N^{\frac{1}{4}}$. Moreover, the entries on the diagonal of each X_i block alone imply the same bound on d. Hence in order to get a bound significantly greater than $N^{\frac{1}{4}}$, one should use as few X-blocks as possible.

2. Not every vector in the Y-block helps to improve the bound on d. In the analysis of the bound we have to satisfy a condition of the form

$$\det(L) \leq e^{m \dim(L)}.$$

   This condition tells us when a vector can be used to improve the bound: Assume this inequality holds and we add a new vector to the lattice bases which contributes to $\det(L)$ by a factor smaller than e^m. Then, the right-hand side increases by a factor of exactly e^m in the inequality since the dimension goes up by 1, but the left-hand side increases by a smaller factor. Hence, we can use a slightly larger value for the upper bound on d, thereby increasing the determinant and the inequality will still hold.

Summarizing the above observation: Every vector which contributes to the determinant with a factor smaller than e^m is *helpful*! Since our new lattice basis construction method produces only triangular bases this means that every vector in the Y-blocks of B_{BD} with diagonal entry smaller than e^m is helpful. Hence, we should use as many of these vectors as possible.

3. In the analysis of the theoretical bounds, we assume that the RSA-modulus N and the lattice dimension $dim(L)$ both go to infinity. However, in order to get good attacks in practice, one should use a parameter choice which gives good results for the size of d while keeping the lattice dimension as small as possible. So, we are also interested in lattice bases with a small number of vectors.

Following the first and third suggestion, one should use a lattice bases where the number of X-blocks is as small as possible. The pattern

$$P_1 := (t+1,\ t,\ t-1,\ \ldots,\ 1)$$

is an input for our lattice basis construction algorithm that satisfies the strictly decreasing pattern restriction and has a minimal number of X-blocks while using as many helpful vectors from the Y-blocks as possible.

On the other hand, we could also take the maximum number of helpful vectors in the Y-block. Let $p_j, j \leq t$, be the number of vectors in the Y_j-block of B_{BD} with diagonal entry smaller than e^m. Then the pattern

$$P_2 := (m+1,\ p_1,\ p_2,\ p_3, \ldots,\ p_t)$$

maximizes the number of helpful vectors from the Y-block. It remains to show that P_2 is also a valid input pattern for our LBC-algorithm, i.e., it is a strictly decreasing pattern. Therefore, we prove the following two properties.

- The diagonal entries $Y_{j,1}, \ldots, Y_{j,m}$ within each Y_j-block have strictly decreasing sizes. This implies that the helpful vectors are the last vectors of each Y_j-block and our LBC-algorithm leaves them in the new basis.

- The pattern P_2 satisfies the strictly decreasing pattern constraint of our LBC-algorithm.

In order to see the first property, notice that by going from the k^{th} vector in a Y_j-block to the $(k+1)^{st}$ vector, the diagonal entries decreases by e and simultaneously increase by a factor of XY. Hence, whenever

$$e > XY,$$

the diagonal entries have strictly decreasing sizes. This condition implies that the upper bound X for d is smaller than $N^{\frac{1}{2}}$, which surely holds for our attacks.

In order to show the second property, we prove that P_2 is a strictly decreasing pattern. To show the inequality $m+1 < p_1$, we observe that the largest vector among the $m+1$ vectors in the Y_1-block has diagonal entry $e^m Y > e^m$. Therefore, it is not a helpful vector.

In order to prove that $p_j < p_{j+1}$, $0 < j < t$, we show that if the k^{th} vector in block Y_j is not a helpful vector, then the $(k+1)^{st}$ vector in the subsequent block Y_{j+1} must also have a diagonal entry that is too large. The diagonal entry of vector $Y_{j,k}$ is $e^{m-k}X^kY^{j+k}$ and the diagonal entry of $Y_{j+1,k+1}$ is $e^{m-k-1}X^{k+1}Y^{j+k+2}$. The second term is greater than the first one whenever

$$XY^2 > e.$$

This is satisfied since $Y^2 > N > e$. Therefore, the number of helpful vectors in the Y_j-blocks strictly decreases with growing j, which implies that P_2 defines a strictly decreasing pattern.

The vectors corresponding to the pattern P_2 form the lattice basis that was used by Boneh and Durfee in order to show the bound of $d < N^{0.292}$. In contrast to our analysis, Boneh and Durfee do not remove columns to make the basis triangular but instead they introduce so-called geometrically progressive matrices in order to show that the determinant mainly depends on the last entries of the chosen vectors.

Our analysis yields an alternative proof of the bound $d < N^{0.292}$. The pattern P_1 achieves a theoretical bound of $d < N^{0.290}$ which is worse than the Boneh-Durfee bound, but from a practical point of view the pattern P_1 is sometimes preferable since it achieves better values for smaller lattice dimension.

Let $B_1(m,t)$ and $B_2(m,t)$ be the lattice bases when our lattice basis construction algorithm is applied to the Boneh-Durfee basis $B_{BD}(m,t)$ with the patterns P_1 and P_2, respectively.

For the rest of this section, we will assume that the size of e is in the order of the size of N. It is straightforward to generalize the following two theorems to arbitrary e. The attacks work for larger d if e is significantly smaller than N. On the other hand, the attacks can be completely counteracted by choosing e suitably large (in the case of the Boneh-Durfee attack: $e > N^{1.875}$).

Theorem 78 *Let $\epsilon > 0$. Furthermore, let the upper bound X for the secret exponent d satisfy*

$$X := N^{\frac{\sqrt{6}-1}{5}-\epsilon}$$

and let $Y := 3N^{\frac{1}{2}}$. For suitably large N, m, t the following holds: On input $B_1(m,t)$ the L^3-algorithm outputs two vectors with norm smaller than $\frac{e^m}{dim(L)}$.

Proof: Let L be the lattice spanned by $B_1(m,t)$ and let n denote the dimension of L. It is not hard to see that $n = (m+1)(t+1)$. According to Theorem 4, we know that the second-to-shortest vector v_2 of an L^3-reduced basis of L satisfies

$$\|v_2\| \leq 2^{\frac{n}{4}} \det(L)^{\frac{1}{n-1}}.$$

In order to apply Howgrave-Graham's theorem (Theorem 14), we need to satisfy the condition

$$2^{\frac{n}{4}} \det(L)^{\frac{1}{n-1}} < \frac{e^m}{\sqrt{n}}.$$

We neglect all terms that do not depend on N. Thus, our condition simplifies to $\det(L) < e^{m(n-1)}$. Furthermore, we set $t := \tau m$. A straightforward calculation shows that

$$\det(L) = \left(e^{3\tau} X^{\tau^3 - 3\tau^2 + 6\tau} Y^{\tau^3 + 3\tau}\right)^{\frac{1}{6}m^3(1+o(1))}$$

We plug in the bounds $X = N^{\frac{\sqrt{6}-1}{5}-\epsilon}$, $Y = 3N^{\frac{1}{2}}$. Using $e = N^{1-o(1)}$ and neglecting low order terms, we obtain the new condition

$$(3 + 2\sqrt{6})\tau^2 + 6(1 - \sqrt{6})\tau - 3(9 + 4\sqrt{6}) \leq 0$$

for $m \to \infty$. The function on the left hand size is minimized at the point $\tau = \frac{3(\sqrt{6}-1)}{2\sqrt{6}+3}$. For this choice of τ, our condition is satisfied. This concludes the proof of Theorem 78. $\Box$

Next, we derive the Boneh-Durfee bound of $d \leq N^{0.292}$.

Theorem 79 *Let $\epsilon > 0$. Furthermore, let the upper bound X for the secret exponent d satisfy*

$$X := N^{1-\sqrt{\frac{1}{2}}-\epsilon}$$

and let $Y := 3N^{\frac{1}{2}}$. For suitably large N, m, t the following holds: On input $B_2(m,t)$ the L^3-algorithm outputs two vectors with norm smaller than $\frac{e^m}{dim(L)}$.

Proof: The proof is similar to the proof of Theorem 78.

Let L be the lattice that is spanned by the basis $B_2(m,t)$ and let $n := \dim(L)$. Furthermore, let $\delta := 1 - \sqrt{\frac{1}{2}} - \epsilon$.

First, we want to optimize the parameter t. Remember that we choose in the pattern P_2 all helpful vectors from the Y-block, i.e., all vectors with diagonal entry at most e^m in the basis matrix $B_2(m,t)$. Observe that the vector $V_{t,m}$ has the diagonal entry $X^m Y^{m+t}$ in $B_2(m,t)$. This vector is helpful as long as $X^m Y^{m+t} \leq e^m$. Neglecting low order terms, this gives as the restriction $t \leq m(1-2\delta)$.

Second, we observe that a vector $Y_{j,k}$ has the diagonal entry $e^{m-k}X^k Y^{k+j}$. A straightforward calculation shows that this vector is useful when $k \geq \frac{j}{1-2\delta}$. Summarizing, we use all vectors $Y_{j,k}$ with $j = 1, \ldots, m(1-2\delta)$ and $k = \frac{j}{1-2\delta}, \ldots, m$ (we neglect roundings) in our lattice basis.
Now we are able to compute the determinant

$$\det(L) = \left(e^{3-2\delta} X^{4(1-\delta)} Y^{4(\delta-1)^2}\right)^{\frac{1}{6}m^3(1-o(1))}.$$

In order to apply Howgrave-Graham's theorem (Theorem 14), we have to satisfy a relation of the form $\det(L) < e^{m(n-1)}$ (see the proof of Theorem 78). It is not hard to see that $m(n-1) = (1-\delta)m^3(1-o(1))$. Using the bounds on X, Y and e and again neglecting low order terms, we obtain the new condition

$$-2\delta^2 + 4\delta - 1 < 0.$$

This condition is satisfied by our choice of δ, which concludes the proof. $\square$

We implemented our new method and carried out several experiments on a Linux-PC with 550 MHz using the pattern $P_1 := (t+1, t, t-1, \ldots, 1)$. The L^3-reduction was done using Victor Shoup's NTL library [63].

In every experiment, we found two vectors with norm smaller than $\frac{e^m}{\sqrt{w}}$. Interestingly in the experiments, we made the following observation:

> The reduced lattice basis contained not only two sufficiently small vectors, but all vectors in the L^3-reduced basis of L had about the same norm.

We find this experimental observation quite surprising. One might be tempted to use this fact in order to extract the secret root (x_0, y_0) by simple linear algebra: Since the basis vectors are linear independent, we can try to solve the system of linear equations in the unknowns $x^i y^j$ by doing Gauss elimination. However, this approach will not work since we cannot use the triangular quadratic lattice basis itself but the corresponding non-triangular basis, where the basis vectors consist of the corresponding reconstruction vectors. Hence, we have more unknowns $x^i y^j$ than equations and we cannot hope to find a unique solution of the linear system. At the moment, we do not know how to use the observation above.

In our experiments, the resultant of two polynomials was computed using the Maple computer algebra system (version 8). The resultant with respect to x was always a polynomial in y, and the root delivered the factorization of N. Our results compare well to those of Boneh and Durfee in the Eurocrypt paper [12]. Boneh and Durfee ran new experiments in [13], but used additional tricks to enlarge d by a few bits:

1. Lattice reduction with Schnorr's block reduction variant [57].
2. Use of Chebychev polynomials (a trick due to Coppersmith [21]).
3. If $\|p(xX, yY)\| < c \cdot e^m/\sqrt{w}$, one knows $|p(x_0, y_0)| < c \cdot e^m$ and $p(x_0, y_0) = 0 \bmod e^m$. Hence, one can guess $\gamma \in (-c, c)$ such that $p(x, y) + \gamma e^m$ satisfies $p(x_0, y_0) + \gamma e^m = 0$ over $\mathbb{Z}$.

These tricks apply to our method as well, but we did not implement them. Comparing instances with the same bit-size of p, q and the same δ as in [12], our algorithm with pattern P_1 was several times faster due to the reduced lattice dimension. The following table contains the running times that we obtained. Where available, we also included the corresponding running times as provided in [12] (these running times were achieved on a 400 MHz SUN workstation).

p, q	δ	m	t	w	our running time	running time in [12]
1000 bits	0.265	4	2	15	6 minutes	45 minutes
3000 bits	0.265	4	2	15	100 minutes	300 minutes
3000 bits	0.269	5	2	18	8 hours	-
500 bits	0.270	6	2	21	19 minutes	-
500 bits	0.274	8	3	36	300 minutes	-
500 bits	0.2765	10	4	55	26 hours	-
500 bits	0.278	11	5	72	6 days	-

In all examples, we chose d uniformly with $\delta \log(N)$ bits until $\log_N(d)$ was equal to δ within precision at least 10^{-4}. The running time measures only the time for L^3-reduction. With growing m and t, the time for resultant computation can take longer than reducing the lattice basis $B(m, t)$.

7.5 A case where the resultant heuristic fails

As mentioned before, if L_{BD} is constructed using only x-shifted polynomials $g_{i,k}$ then the Boneh-Durfee method always failed in our experiments. More precisely, the polynomials we obtained from the two shortest vectors in an L^3-reduced basis for L_{BD} led to two polynomials whose resultant with respect to x was identically 0. We want to explain this phenomenon.

Using the construction of Section 7.3 for $B_{BD}(m,0)$ with the pattern $P=(1)$, the lattice L consists only of the vectors in the block X_m with the columns in $X^{(m)}$. A simple determinant computation shows, that for every $l \leq m$ there is a linear combination of vectors in block X_l that is shorter than e^m provided $\delta < 0.25$.

Moreover, unless a combination of vectors in block X_l is much shorter than e^m (according to Lemma 76 it must be of size $\mathcal{O}(\frac{e^m}{XY})$ in order to be helpful, since the entries in the $X^{(m-1)}$ column block are already of size $\mathcal{O}(\frac{e^m}{XY})$), combinations of vectors from different blocks X_l, X_j cannot be shorter than vectors obtained as combinations of vectors from a single block. Although not a rigorous proof, this explains the following observation. In our experiments every vector in an L^3-reduced basis for the original $B_{BD}(m,0)$ was a combination of basis vectors from a single block X_l. In fact, the following was true for arbitrary L_{BD}, even those constructed using y-shifts: Every vector in an L^3-reduced basis for L_{BD} that depended only on basis vectors in the X-block was a combination of basis vectors from a single block X_l.

Now assume that we have a vector that is a linear combination of a single X^l block with a sufficiently small norm satisfying the second condition of Howgrave-Graham's theorem, e.g. the corresponding polynomial $p(x,y)$ has the root (x_0,y_0) over the integers. The following theorem tells us how to extract the secret root (x_0,y_0) and the factorization of N in time polynomial in the bit-size of N.

Theorem 80 *Let $p(x,y)$ be a polynomial that is a non-zero linear combination of the X_l-block with $p(x_0,y_0)=0$. Then the factorization of N can be found in polynomial time.*

Proof: Write

$$p(x,y) = \sum_{i=0}^{l} c_i x^{l-i} f^i(x,y) e^{m-i}.$$

Now we evaluate $p(x,y)$ at the point (x_0,y_0). Since $f(x_0,y_0) = k(A-(p+q))-1 = -ed$, we obtain the equation

$$p(x_0,y_0) = e^m \sum_{i=0}^{l} c_i x_0^{l-i} d^i = 0$$

in the unknown parameters x_0 and d. Hence the polynomial

$$q(x,z) = c_0 x^l + c_1 x^{l-1} z + c_2 x^{l-2} z^2 + \cdots + c_l z^l$$

must have the root $(x_0,z_0)=(k,d)$ over the integers. Since

$$c_0 k^l = -d(c_1 k^{l-1} + c_2 k^{l-2} d + c_l d^{l-1})$$

and $\gcd(d,k)=1$, we see that d must divide c_0. Analogously, one can show that k must divide c_l. On the other hand we may not be able to recover the unknowns d, k in polynomial time by factoring c_0, c_l.

Therefore, we propose another way to extract the secrets d and k. Observe that $q(x,z)$ does not only contain the root (x_0, z_0) but also every multiple (ax_0, az_0), $a \in \mathbb{Z}$, as a root. Hence the irreducible polynomial $h(x,z) = z_0 x - x_0 z$, which contains all the roots (ax_0, az_0), must divide $q(x,z)$. Therefore we can obtain an integer multiple $b \cdot h(x,z) = h_1 x + h_2 z$ of h by factoring $q(x,z)$ into irreducible polynomials over $\mathbb{Q}[x,z]$. Since $\gcd(x_0, z_0) = \gcd(k,d) = 1$, we obtain $d = \frac{h_1}{\gcd(h_1,h_2)}$ and $k = -\frac{h_2}{\gcd(h_1,h_2)}$.

Finally, the knowledge of d and k gives as the term $p+q$ from which we derive the factorization of N. □

In the proof of Theorem 80, we showed that one can extract from every linear combination of vectors of an X-block a linear bivariate polynomial $h(x,z)$ that yields the secret parameters k and d. Since this holds for every X-block, we conclude in the following corollary that vectors constructed from different X-blocks must share a non-trivial greatest common divisor polynomial.

This implies that the resultant heuristic does not apply for our special settings, since the resultants all vanish. However, on the other hand this does not imply that Coppersmith's method fails in general for this case, since we can extract the roots by applying Theorem 80.

Hence, as a positive side-effect of the failure of the resultant heuristic we obtain a provable method that extracts the secrets of a bivariate polynomial of a special form. Now let us prove that the resultant heuristic indeed cannot be applied for the case mentioned above.

Corollary 81 *Let $p_1(x,y)$ and $p_2(x,y)$ be polynomials that are non-zero linear combinations of the X_{l_1}-block and X_{l_2}-block, respectively. If $p_1(x_0,y_0) = p_2(x_0,y_0) = 0$ then p_1 and p_2 share a non-trivial greatest common divisor.*

Proof: We follow the reasoning of the proof of Theorem 80: By substituting $ez = f(x,y)$ we can show that the polynomial $h(x,z) = dx - kz$ divides both polynomial p_1 and p_2. But then p_1 and p_2 share the polynomial $dx - \frac{k}{e} f(x,y)$ as a common divisor. This polynomial must also divide $\gcd(p_1, p_2)$ which concludes the proof. □

Bibliography

[1] T. M. Apostol, *Introduction to analytic number theory*, Springer-Verlag, 1980

[2] M. Bellare, P. Rogaway, “Optimal Asymmetric Encryption”, Advances in Cryptology – Eurocrypt ’94, Lecture Notes in Computer Science Vol. 950, Springer-Verlag, pp. 92–111, 1994

[3] D. Bleichenbacher, “On the Security of the KMOV public key cryptosystem”, Advances in Cryptology – Crypto ’97, Lecture Notes in Computer Science Vol. 1294, Springer-Verlag, pp. 235–248, 1997

[4] J. Blömer, “Closest vectors, successive minima, and dual HKZ-bases of lattices”, Proceedings of 17th ICALP, Lecture Notes in Computer Science Vol. 1853, pp. 248–259, 2000.

[5] J. Blömer, A. May, “New Partial Key Exposure Attacks on RSA”, Advances in Cryptology – Crypto 2003, Lecture Notes in Computer Science Vol. 2729, pp. 27–43, Springer-Verlag, 2003

[6] J. Blömer, A. May, “A Generalized Wiener Attack on RSA”, Practice and Theory in Public Key Cryptography – PKC 2004, Lecture Notes in Computer Science Vol. 2947, pp. 1–13, Springer-Verlag, 2004

[7] J. Blömer, A. May, “Low Secret Exponent RSA Revisited”, Cryptography and Lattice Conference (CaLC 2001), Lecture Notes in Computer Science Volume 2146, Springer-Verlag, pp. 4–19, 2001.

[8] J. Blömer, M. Otto, personal communication

[9] D. Boneh, “Twenty years of attacks on the RSA cryptosystem”, Notices of the AMS, 1999

[10] D. Boneh, “Simplified OAEP for the RSA and Rabin Functions”, Advances in Cryptology – Crypto 2001, Lecture Notes in Computer Science Vol. 2139, pp. 275–291, Springer-Verlag, 2001

[11] D. Boneh, R. DeMillo, R. Lipton, "On the importance of checking cryptographic protocols for faults", Advances in Cryptology – Eurocrypt'97, Lecture Notes in Computer Science Vol. 1233, Springer-Verlag, pp. 37–51, 1997.

[12] D. Boneh, G. Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$", Advances in Cryptology – Eurocrypt'99, Lecture Notes in Computer Science Vol. 1592, Springer-Verlag, pp. 1–11, 1999.

[13] D. Boneh, G. Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$", IEEE Trans. on Information Theory, Vol. 46(4), pp. 1339–1349, 2000

[14] D. Boneh, G. Durfee, Y. Frankel, "An attack on RSA given a small fraction of the private key bits", Advances in Cryptology – Asiacrypt '98, Lecture Notes in Computer Science Vol. 1514, Springer-Verlag, pp. 25–34, 1998

[15] D. Boneh, G. Durfee, Y. Frankel, "Exposing an RSA Private Key Given a Small Fraction of its Bits", Full version of the work from Asiacrypt'98, available at `http://crypto.stanford.edu/~dabo/abstracts/bits_of_d.html`, 1998

[16] D. Boneh, G. Durfee, and N. Howgrave-Graham, "Factoring $N = p^r q$ for large r", Advances in Cryptology – Crypto '99, Lecture Notes in Computer Science Vol. 1666, Springer-Verlag, pp. 326–337, 1999

[17] D. Boneh, Venkatesan, " Breaking RSA may not be equivalent to factoring", Advances in Cryptology – Eurocrypt '98, Lecture Notes in Computer Science Vol. 1233, Springer-Verlag, pp. 59–71, 1998

[18] D. Coppersmith, "Finding a Small Root of a Univariate Modular Equation", Advances in Cryptology – Eurocrypt '96, Lecture Notes in Computer Science Vol. 1070, Springer-Verlag, pp. 155–165, 1996

[19] D. Coppersmith, "Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known", Advances in Cryptology – Eurocrypt '96, Lecture Notes in Computer Science Vol. 1070, Springer-Verlag, pp. 178–189, 1996

[20] D. Coppersmith, "Small solutions to polynomial equations and low exponent vulnerabilities", Journal of Cryptology, Vol. 10(4), pp. 223–260, 1997.

[21] D. Coppersmith, "Finding Small Solutions to Small Degree Polynomials", Cryptography and Lattice Conference (CaLC 2001), Lecture Notes in Computer Science Volume 2146, Springer-Verlag, pp. 20–31, 2001.

[22] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer-Verlag, 1996

[23] D. Cox, J. Little, D. O'Shea, *Ideals, Varieties and Algorithms*, Springer-Verlag, 1992

[24] R. Cramer, V. Shoup, "A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack", Advances in Cryptology – Crypto '98, Lecture Notes in Computer Science Vol. 1462, Springer-Verlag, pp. 13–25, 1998

[25] C. Crépeau, A. Slakmon, "Simple Backdoors for RSA Key Generation", Topics in Cryptology – CT-RSA 2003, Lecture Notes in Computer Science Vol. 2612, pp. 403–416, Springer-Verlag, 2003

[26] W. Diffie, M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory Vol. 22, pp. 644–654, 1976

[27] J. F. Dhem, F. Koeune, P. A. Leroux, P. Mestre, J. J. Quisquater, and J. L. Willems, "A practical implementation of the timing attack", Proceedings of CARDIS 98 – Third Smart Card Research and Advanced Application Conference, 1998

[28] G. Durfee, P. Nguyen, "Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt '99", Advances in Cryptology – Asiacrypt 2000, Lecture Notes in Computer Science Vol. 1976, Springer, pp. 14–29, 2000

[29] T. El Gamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", Advances in Cryptology – Crypto '84, Lecture Notes in Computer Science Vol. 196, Springer-Verlag, pp. 10–18, 1984

[30] A. Fujioke, T. Okamoto, Miyaguchi, "ESIGN: An Efficient Digital Signature Implementation for Smartcards", Advances in Cryptology – Eurocrypt '91, Lecture Notes in Computer Science Vol. 547, Springer-Verlag, pp. 446-457, 1991

[31] E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern, "RSA-OAEP Is Secure under the RSA Assumption", Advances in Cryptology – Crypto 2001, Lecture Notes in Computer Science Vol. 2139, Springer-Verlag, pp. 260–274, 2001

[32] J. von zur Gathen, J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, Cambridge, 1999

[33] M. Gruber, C.G. Lekkerkerker, *Geometry of Numbers*, North-Holland, 1987

[34] G. H. Hardy, E. M. Wright, *Introduction to the Theory of Numbers*, Oxford University Press, 1979.

[35] N. Howgrave-Graham, "Finding small roots of univariate modular equations revisited", Proceedings of Cryptography and Coding, Lecture Notes in Computer Science Vol. 1355, Springer-Verlag, pp. 131–142, 1997

[36] N. Howgrave-Graham, “Approximate Integer Common Divisors”, Cryptography and Lattice Conference (CaLC 2001), Lecture Notes in Computer Science Vol. 2146, Springer-Verlag, pp. 51–66, 2001

[37] C. Jutla, “On finding small solutions of modular multivariate polynomial equations”, Advances in Cryptology – Eurocrypt ’98, Lecture Notes in Computer Science Vol. 1403, Springer-Verlag, pp. 158–170, 1998

[38] Erich Kaltofen, “Polynomial-Time Reductions from Multivariate to Bi- and Univariate Integral Polynomial Factorization”, SIAM Journal on Computing Vol. 14(2), pp. 469–489, 1985

[39] Knuth, *The Art of Computer Programming*, Third Edition, Addison-Wesley, 1997

[40] N. Koblitz, *A course in number theory and cryptography*, Springer-Verlag, 1994

[41] P. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems”, Advances in Cryptology – Crypto ’96, Lecture Notes in Computer Science Vol. 1109, Springer Verlag, pp. 104–113, 1996

[42] P. Kocher, J. Jaffe and B. Jun, “Differential power analysis”, Advances in Cryptology – Crypto ’99, Lecture Notes in Computer Science Vol. 1666, Springer-Verlag, pp. 388–397, 1999

[43] H. W. Lenstra, “Factoring Integers with Elliptic Curves”, Mathematische Annalen, Vol. 126, pp. 649–673, 1987

[44] A. K. Lenstra, H. W. Lenstra, and L. Lovász, ”Factoring polynomials with rational coefficients,” Mathematische Annalen, Vol. 261, pp. 513–534, 1982

[45] A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse, and J. M. Pollard, “The number field sieve”, In Proceedings of the 22nd Annual ACM Symposium on the Theory of Computation, pages 564–572, 1990

[46] L. Lovász, *An Algorithmic Theory of Numbers, Graphs and Convexity*, Conference Series in Applied Mathematics, SIAM, 1986

[47] U. Maurer and S. Wolf, “Diffie-Hellman Oracles”, Advances in Cryptology – Crypto ’96, Lecture Notes in Computer Science Vol. 1109, Springer-Verlag, pp. 268–282, 1996

[48] A. May, “Cryptanalysis of Unbalanced RSA with Small CRT-Exponent”, Advances in Cryptology – Crypto 2002, Lecture Notes in Computer Science Vol. 2442, Springer-Verlag, pp. 242–256, 2002

[49] A. May, "Secret Exponent Attacks on RSA-type Schemes with Moduli $N = p^r q$", Practice and Theory in Public Key Cryptography – PKC 2004, Lecture Notes in Computer Science Vol. 2947, Springer-Verlag, pp. 218–230, 2004

[50] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996

[51] H. Minkowski, *Geometrie der Zahlen*, Teubner Verlag, 1912

[52] N. Modadugu, D. Boneh, M. Kim, "Generating RSA Keys on a Handheld Using an Untrusted Server", Progress in Cryptology - Indocrypt 2000, Lecture Notes in Computer Science Vol. 1977, Springer-Verlag, pp. 271-282, 2000

[53] P. Nguyen, J. Stern, "Lattice Reduction in Cryptology: An Update", Algorithmic Number Theory Symposium ANTS-IV, pp. 85-112, 2000

[54] T. Okamoto, S. Uchiyama, "A New Public-Key Cryptosystem as Secure as Factoring", Advances in Cryptology – Eurocrypt '98, Lecture Notes in Computer Science Vol. 1403, Springer-Verlag, pp. 308–318, 1998

[55] J.-J. Quisquater, C. Couvreur, "Fast decipherment algorithm for RSA public-key cryptosystem", Electronic Letters 18 (21), pp. 905–907, 1982

[56] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, Vol. 21(2), pp.120–126, 1978

[57] C.P. Schnorr, "A hierarchy of polynomial time lattice basis reduction algorithms", Theoretical Computer Science, Vol. 53, pp.201–224, 1987

[58] C.P. Schnorr, "Gittertheorie und algorithmische Geometrie", Vorlesungsskript Universität Frankfurt, available at `http://ismi.math.uni-frankfurt.de/schnorr/lecturenotes/schnorr.gitter.ps`, 1998

[59] I.R. Shafarevich, *Basic Algebraic Geometry*, Springer-Verlag, 1994

[60] A. Shamir, "RSA for paranoids", CryptoBytes Vol. 1(3), pp. 1-4, 1995

[61] P. W. Shor, "Algorithms for quantum computation: Discrete log and factoring", In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pages 124–134, 1994

[62] V. Shoup, "OAEP Reconsidered", Advances in Cryptology – Crypto 2001, Lecture Notes in Computer Science Vol. 2139, Springer-Verlag, pp. 239–259, 1998

[63] V. Shoup, NTL: A Library for doing Number Theory, online available at `http://www.shoup.net/ntl/index.html`

[64] C.L. Siegel, *Lectures on the Geometry of Numbers*, Springer-Verlag, 1989

[65] D. Stinson, *Cryptography Theory and Practice*, Second Edition, CRC Press, 2002

[66] H.-M. Sun, W.-C. Yang and C.-S. Laih, "On the design of RSA with short secret exponent", Advances in Cryptology - Asiacrypt '99, Lecture Notes in Computer Science Vol. 1716, Springer Verlag, pp. 150–164, 1999

[67] T. Takagi, "Fast RSA-type cryptosystem modulo $p^k q$", Advances in Cryptology – Crypto '98, Lecture Notes in Computer Science Vol. 1462, Springer-Verlag, pp. 318–326, 1998

[68] E. Verheul, H. van Tilborg, "Cryptanalysis of less short RSA secret exponents", Applicable Algebra in Engineering, Communication and Computing, Vol. 8, Springer-Verlag, pp. 425–435, 1997

[69] S. A. Vanstone, R. J. Zuccherato, "Short RSA Keys and Their Generation", Journal of Cryptology 8(2), pp. 101–114, 1995

[70] B. de Weger, "Cryptanalysis of RSA with small prime difference", Applicable Algebra in Engineering, Communication and Computing ,Vol. 13(1), Springer-Verlag, pp. 17–28, 2002

[71] M. Wiener, "Cryptanalysis of short RSA secret exponents", IEEE Transactions on Information Theory, Vol. 36, pp. 553–558, 1990

[72] C. Yap, "Fundamental Problems in Algorithmic Algebra", Oxford University Press, 1999

[73] S.-M. Yen, S. Kim, S. Lim, S. Moon, "Speedup with Residue Number System Immune against Hardware Fault Cryptanalysis", 4^{th} International Conference on Information Security and Cryptology, Lecture Notes in Computer Science Vol. 2288, Springer-Verlag, pp. 397–413, 2001.

[74] S.-M. Yen, S. Kim, S. Lim, S. Moon, "RSA Speedup with Chinese Remainder Theorem Immune against Hardware Fault Cryptanalysis", IEEE Transactions on Computers, Vol. 52(4), pp. 461–472, 2003