University of Zurich
Institute of Mathematics

Master Thesis

The ROCA Vulnerability: A Study on Coppersmith's Algorithm and its Applications

Author:
Tatjana Bossalini

Supervisor:
Prof. Dr. Joachim Rosenthal

September 2023

Acknowledgements

I owe many thanks and a great deal of gratitude to numerous people without whom, I would not have been able to accomplish this.

I would like to start by expressing my deepest and most sincere gratitude to my supervisor, Prof. Dr. Joachim Rosenthal. Thanks to his unwavering enthusiasm and passion during the lectures, I discovered my love for algebra and cryptography, inspiring me to embark on the journey of this thesis. I am also very grateful to Abhinaba for helping me with corrections and comments despite not being my co-supervisor.

I could not have even started this journey without the support and sacrifices of my whole family. In particular, I want to thank my sister Aisha for always believing in me and for her constant presence through thick and thin.

My heartfelt thanks go to all the incredible friends I have had the privilege of making within the mathematics community.
Thank you, Diana, for supporting me since the start. I still remember when I was thinking about withdrawing from university during my very first semester. Without your encouragement, this page wouldn't even exist.
A big thank you goes to Renzo and his patience for teaching and guiding me through the intricate world of programming with Mathematica.
Iana, Dada, and Thomas, I am deeply grateful for the laughs and chill moments. They were among the best solutions that helped me manage the challenges of university.
Thank you, Giulia, Kata, Valo, Mattia, Elia, Dadenano, and Fede, for the good times spent together: playing Tichu (the best card game ever), volleyball, dinners, parties, etc. It was all fun because of you. Also, thank you for being the kindest and most generous people I have ever met. It was a relief and a blessing to know that I could always count on you for anything.
It would be negligent of me to omit acknowledging my fellow companions in university misadventures: Chri, Timo, Ely, Nico, Fabb, Piatto, all members of 'Mate UZH' and 'Beachwolley ZH', the intruder from ETH Dani, and the non-mathematician extensions Crotta, Chris, Cesare, Lucri, Raph, Filo and Gabriel.

Last but not least, I want to express my gratitude to Lorenzo, Davide, and Sofia, who have been like a second family during the last few years. Returning home after the lessons, knowing that I would have had the opportunity to be in your company, brought joy and solace to my life.

To all of you, thank you for your friendship, for your encouraging and motivating words, and for being such wonderful people. You will always have a special place in my heart, as well as in this thesis. Lastly, thank you for helping me get to this point and develop into the person I am today, a person I am proud of.

Thank you

Abstract

The search for strong security and effective algorithms is never-ending in the field of modern cryptography. Cryptography is the foundation of trust in the digital era, covering everything from our private communications to financial transactions and essential infrastructure. It enables us to share sensitive information and communicate with confidence, knowing that our data remains confidential and secured.
This thesis presents an introduction to Coppersmith's algorithm. We start by laying the theoretical groundwork for three variants: univariate modular, bivariate integer, and multivariate modular case. Subsequently, we proceed to analyze the diverse range of applications that can be derived from these findings, including but not limited to the use of stereotyped messages, random padding for two messages, factoring with incomplete information, and the Chinese Remaindering with Errors problem. We developed and implemented code examples for each scenario, providing a tangible demonstration of the algorithm's efficiency. Finally, we discuss the ROCA vulnerability that was found in Estonia and its effects on the nation. This weakness allows an attacker to factorize the RSA modulus and compromise the security of the affected systems. The Estonian example emphasizes the significance of using secure implementation techniques, being careful when choosing random number generators, and being cautious when picking cryptographic libraries.

Contents

1 Introduction **1**

2 Preliminaries **3**
2.1 Public-Key Cryptography 3
2.1.1 RSA Cryptosystem 4
2.2 Lattices 4
2.2.1 Shortest Vector Problem and basis reduction techniques 7

3 Coppersmith's Method **13**
3.1 Univariate modular case 13
3.2 Bivariate integer case 20
3.3 Multivariate modular case 24
3.4 Notes on Complexity 24

4 Applications **25**
4.1 Stereotyped messages 25
4.2 Random Padding for Two Messages 28
4.3 Factoring with partial knowledge 30
4.4 Factoring $p^r q$ 33
4.5 Chinese Remaindering with Errors 35

5 The ROCA Vulnerability **41**
5.1 Technical details and Key generation process 41
5.1.1 Simulations 45
5.2 ROCA in Estonia 45
5.3 Prevention against future Attacks 46
5.4 Other global impacts 48

6 Conclusion **51**

A List of codes **53**

Acronyms **63**

Bibliography **64**

Chapter 1

Introduction

Cryptography has a rich history that can already be noticed in ancient times. The first evidences of cryptography can be traced back to ancient Egypt around 1900 BC. The precise details of their methods are still a mystery, but it is obvious that they already recognized the need for secrecy and to protect information. As time progressed, more advanced and sophisticated cryptographic methods were born. Ancient Greeks are known for using the *Scytale* - a tool consisting in a leather strip with a written message to wrap around a cylinder to decrypt it - to communicate in a confidential manner [20]. However, it is only around the 20^{th} century that a more modern and rigorous cryptography began to be used. With the advent of modern computing and telecommunications, new threats arose and the need for secure communication became even more pressing. Various cryptographic schemes often hinges on the presumed difficulty of certain mathematical problems. Among these problems, the factorization of large integers and the computation of discrete logarithms are of paramount importance. These mathematical challenges form the basis of widely used encryption and signature schemes, making their efficient solution a topic of profound interest.

Coppersmith's method, introduced by Don Coppersmith in 1996 [13], has emerged as a powerful algorithmic tool for solving certain types of polynomial equations over finite fields. Originally designed for the cryptanalysis of specific cryptographic protocols, Coppersmith's method has found applications in diverse areas, ranging from integer factorization to lattice-based attacks. The method's elegant mathematical foundation and its implications in cryptanalysis have garnered considerable attention from researchers and practitioners alike.

In this thesis, we will study Coppersmith's method, from its theoretical foundations to its real-world applications. By examining the algorithm, its strengths, and its limitations, we aim to highlight the importance of secure information properly and what can be the worst consequences.

The subsequent chapters of this thesis are organized as follows: Chapter 2 presents a review of foundamental and relevant mathematical concepts such as RSA cryptosystems and lattices. In Chapter 3, we introduce and explain the theoretical underpinnings of Coppersmith's by studying different cases: from the univariate modular case to the

extension on more variables. Chapter 4 focuses on the numerous applications of the method. Chapter 5 explores the discovery of the ROCA vulnerability, an attack exploiting Coppersmith's results and the consequences. Lastly, Chapter 6 offers a conclusion and suggests potential directions for further research in this field.

Chapter 2

Preliminaries

In this section, we provide an introduction to some fundamental mathematical concepts which might be useful to facilitate a better comprehension of the contents of this thesis. The information presented here predominantly originates from the lecture notes I took while attending the university courses offered by Prof. Dr. J. Rosenthal and which are summarized in [41].

2.1 Public-Key Cryptography

Prior to the mid-1970s, symmetric cryptography, which entails using a single key for both encryption and decryption, was the dominant form of cryptography. However, this approach was not particularly secure since the key had to be securely exchanged between the parties before any encryption could occur. This could be challenging and risky, especially when dealing with a large number of users. A significant shift in cryptography history began in 1976, when the public-key cryptosystem "Diffie-Hellman key exchange" was published [21]. This became soon the most used cryptosystem of modern times.

Here is a typical example to understand how public-key cryptography works: Bob wants to send a message and share information to Alice over an insecure channel. They agree on a prime number p and a common element $g \in G$ - where G denotes the multiplicative group of integers modulo p - which will be the public key. Then, Alice chooses a random number $a \in G$ which she will keep secret and sends $A = g^a \mod p$. Bob does the same with a secret number $b \in G$: he computes $B = g^b \mod p$ and sends it to Alice. Finally, Alice computes $B^a \mod p$ and Bob $A^b \mod p$. They share the same result since

$$A^b \mod p = (g^a)^b \mod p = g^{ab} \mod p = g^{ba} \mod p = (g^b)^a \mod p = B^a \mod p.$$

This system is considered to be more secure than previous ones since by using a pair of keys, it allows for secure key exchange and digital signatures, and provides authentication and scalability.

2.1.1 RSA Cryptosystem

RSA is one of the oldest public-key cryptography system. The system takes name from its inventors Rivest, Shamir, and Adleman who first published the algorithm in 1977 [40]. The security of RSA is based on the difficulty of factoring large composite numbers. In RSA, each user has a public key and a private key. The public key is used for encryption, while the private key is used for decryption. To create the keys, one needs to choose two large prime numbers p and q and compute their product $n = pq$. The totient function of n is then calculated as $\phi(n) = (p-1)(q-1)$. The public key is a pair of integers (n, e), where e is a randomly chosen integer coprime to $\phi(n)$. The private key is a pair of integers (n, d), where d is the modular inverse of e modulo $\phi(n)$ and which can be computed using Euclidean's algorithm since

$$de + b\phi(n) = 1.$$

The message m is encrypted by Alice by computing $c = m^e \mod n$. Bob can then decrypt the message using his private key d:

$$c^d = m^{ed} = m \mod n.$$

Knowing p and q is polynomially equivalent to knowing n and $\phi(n)$ which means that one can break the system with such information. In fact, assume one manages to discover the value of p. Then, they can easily find q since $q = n/p$. Thanks to this information, one proceeds by computing $\phi(n)$ with the above formula. It remains to find d, but this is also immediate: using the fact that $e \cdot d \equiv 1 \mod \phi(n)$ and extended Euclidean algorithm, its value is acknowledged.

As mentioned above, the security of RSA relies on the fact that it is computationally infeasible to factor n into its prime factors p and q, given only the public key (n, e), making it one of the most secure cryptosystems employed worldwide. Furthermore, RSA is resistant to attacks using brute force methods because there are simply too many possible keys to try. However, even though RSA is one of the most reliable cryptosystems, it is not immune to attacks. In this thesis we will discuss some situations where RSA can show weaknesses and what consequences this may imply.

2.2 Lattices

Lattices are mathematical structures that have become increasingly important in many areas of mathematics. In particular, they find many applications in computer science, and cryptography. They can be seen as discrete sets of points in Euclidean space that are arranged in a regular pattern (see fig. 2.1).

The basic idea is to use the hardness of certain computational problems on lattices to build cryptographic primitives such as digital signatures, encryption schemes, and key exchange protocols.

Definition 2.1. Let $\{b_1, \ldots, b_k\} \subseteq \mathbb{R}^n$ be a set of k linearly independent vectors. Then

$$\Lambda = \{\sum_{i=1}^{k} \lambda_i b_i | \lambda_i \in \mathbb{Z}, i = 1, \ldots, k\}$$

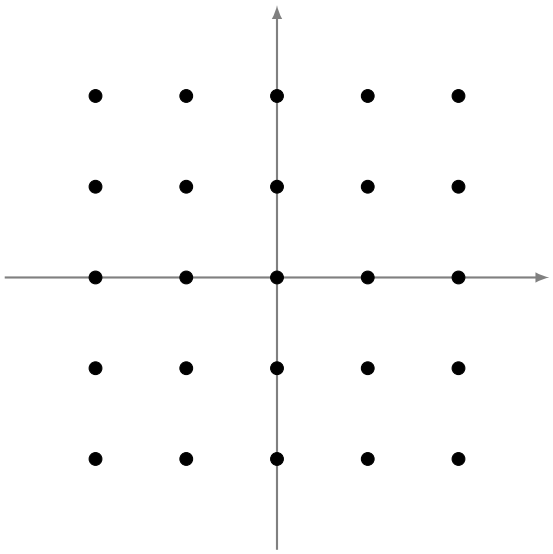

Figure 2.1: An example of lattice corresponding to $\mathbb{Z}^2$.

is called a k-dimensional lattice.

The vectors $b_1, \ldots, b_k$ form a basis for the lattice which is not unique; there can be different bases which generate the same lattice (see fig. 2.3).

A lattice can be represented as a matrix B, called a generator matrix, whose rows consist of the basis vectors. Any element of Λ has then form $(\lambda_1, \ldots, \lambda_k)B$. Working with the lattice as a matrix can be useful for various lattice computations and analyses, for this reason it is important to choose an appropriate and convenient basis.

Notation 2.2. We denote by $\Lambda(B)$ the lattice generated by the matrix B.

Lemma 2.3. *B and B' define the same lattice Λ if and only if there exists a $k \times k$ unimodular matrix U s.t. $B' = UB$.*

Remark 2.4. A unimodular matrix is a square integer matrix whose determinant is equal to ± 1.

Proof.

$\Rightarrow$ Let $\Lambda(B) = \Lambda(B')$. Then, $b_1, \ldots, b_k \in \Lambda(B')$ and $b'_1, \ldots, b'_k \in \Lambda(B)$.

By the definition of lattice have $B = UB'$ and analogously, $B' = U'B$ where $U, U' \in \mathbb{Z}^{k \times k}$. Combining the two equations we get

$$B = UU'B,$$

which is equivalent to

$$0 = UU'B - B = (UU' - \mathbf{I_k})B.$$

Since the rows of B are linearly independent, it must hold $UU' - \mathbf{I_k} = 0$ and hence $UU' = \mathbf{I_k}$. Finally, since $U, U' \in \mathbb{Z}^{k \times k}$ and

$$\det(UU') = \det(U)\det(U') = \det(\mathbf{I_k}) = 1,$$

we conclude that $\det(U) = \det(U') = \pm 1$.

$\Leftarrow$ Suppose now that $B = UB'$ for a unimodular matrix $U \in \mathbb{Z}^{k \times k}$. Since it is unimodular, U is invertible and hence

$$B' = U^{-1}B.$$

This implies that $\Lambda(B') \subseteq \Lambda(B)$ and, by symmetry, $\Lambda(B') \subseteq \Lambda(B)$.

Therefore

$$\Lambda(B) = \Lambda(B'),$$

which completes the proof.

□

Corollary 2.5. *The following operations on a generator matrix B of a $k-$dimensional lattice Λ do not modify the lattice:*

1. *Swapping two rows;*
2. *Multiplying a row by -1;*
3. *Adding a linear combination of any basis vector to a row.*

Definition 2.6 (Fundamental region)**.** The fundamental region of an $n-$dimensional lattice Λ with basis $b_1, \ldots, b_n$ is the parallelepiped

$$\mathcal{P} = \left\{ \sum_{i=1}^{n} b_i \;\middle|\; 0 < b_i < 1 \; \forall i = 1, \ldots n \right\}.$$

Definition 2.7. If the $k \times n$ matrix B defines the lattice $\Lambda = rowsp_{\mathbb{Z}}(B)$, one defines the volume of the fundamental region

$$vol(\Lambda) = \sqrt{det(BB^T)}.$$

In the case B is non-singular, we have

$$vol(\Lambda) = |det(B)|.$$

Furthermore, one can define the density of Λ as

$$d(\Lambda) = \frac{\pi r^2}{vol(\Lambda)}.$$

where r denotes the radius of a $k-$dimensional ball.

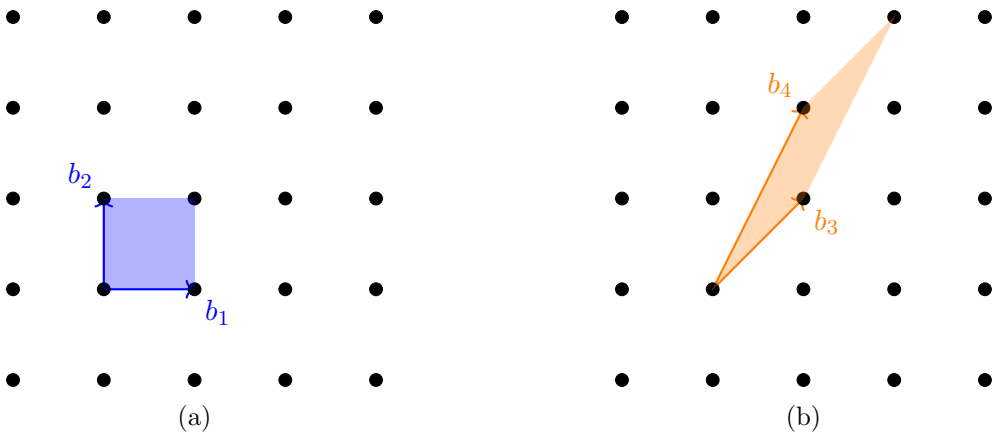

Figure 2.2: The same lattice generated by basis vectors (a) $b_1 = (1, 0)$ and $b_2 = (0, 1)$, and (b) $b_3 = (1, 1)$ and $b_4 = (1, 2)$. The shaded area is the determinant w.r.t. the chosen basis.

Definition 2.8. Let $\Lambda \subseteq \mathbb{R}^m$ be a Lattice with rank n and $\|.\|$ an arbitrary norm. The successive minima $\lambda_1,\ldots,\lambda_n$ w.r.t. the norm $\|.\|$ are defined as follows:

$$\lambda_i = \lambda_i(\Lambda) = \inf\left\{ r > 0 : \begin{array}{l} \exists\, i \text{ linear independent vectors } c_1, ..., c_i \in \Lambda \\ \text{with } \|c_j\| \leq r \text{ for } j = 1, 2, \ldots, i \end{array} \right\}, \quad i = 1, \ldots, n.$$

Remark 2.9. It follows $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

2.2.1 Shortest Vector Problem and basis reduction techniques

The Shortest Vector Problem (SVP) is an important computational problem in lattice-based cryptography and mathematical research. It involves finding the shortest nonzero vector in a given lattice. This problem is NP-hard, which means that it is computationally infeasible to solve in polynomial time using any known algorithm [4, 43]. Therefore, there are many researches which focuses on finding approximation algorithms that can find vectors that are relatively close to the shortest vector in a given lattice.

One of the main applications of the shortest vector problem is in cryptography, where it is used as the foundation of many cryptographic schemes. In particular, the hardness of SVP is used to ensure the security of these cryptographic schemes, making it crucial to study and understand this problem.

Definition 2.10 (Shortest Vector Problem)**.** Let $\Lambda \subset \mathbb{R}^n$ be a non-trivial lattice and B its basis. Given a norm $\|.\|$, find a non-zero lattice vector $u \in \Lambda$ such that

$$\|u\| = \min\{\|v\| \mid v \in \Lambda, v \neq 0\}.$$

Remark 2.11. Notice that the shortest vector u is not unique. An obvious example is the vector $-u$, which possesses the same length but points in the opposite direction. However, there might be other different vectors which share the same norm and are linearly independent from u.

We'll see now some techniques to find relatively short vectors in a lattice. To begin with, we find a nice bound for the vectors in a lattice in the following theorem.

Theorem 2.12 (Minkowski)**.** *Let $\Lambda \subseteq \mathbb{R}^n$ a n-dimensional lattice. Then there exist $b \in \Lambda \setminus \{0\}$ s.t. $\|b\| \leq \sqrt{n}\det(\Lambda)^{\frac{1}{n}}$, where $\|b\|$ denotes the Euclidean norm.*

A way to simplify computational problems related to SVP is finding a smaller basis for a lattice. The idea of basis reduction in an n-dimensional lattice is to find the orthogonal basis based on the given basis. A good way to construct such a basis is the Gram-Schmidt orthogonalization process.

Theorem 2.13 (Gram-Schmidt Orthogonalization)**.** *Let $\{b_1, \ldots, b_n\}$ be the basis of an n-dimensional lattice Λ. Let*

$$b_i^* = b_i - \sum_{i \leq n} \mu_{ij} v_j^*, \qquad \text{where} \qquad \mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}.$$

Then, $\{b_1^*, \ldots, b_n^*\}$ is an orthogonal basis for Λ.

The Gram-Schmidt Orthogonalization process is already quite close to a basis reduction algorithm. The only problem is that the resulting vectors are not necessarily in the lattice. Adding some small conditions, it is possible to create an algorithm that solves this problem and achieves the size-reduction notion.

Definition 2.14 (LLL-reduced basis)**.** Let $\{b_1, \ldots, b_k\}$ be the basis of a k-dimensional lattice and $\{b_1^*, \ldots, b_k^*\}$ the Gram-Schmidt orthogonal basis i.e. $b_i^* = b_i - \sum \mu_{ij} v_j^*$ where $\mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$. Then, we say that $\{b_1, \ldots, b_k\}$ is LLL-reduced with parameter δ, $\frac{1}{4} < \delta < 1$ if the following conditions hold:

1. $\|\mu_{ij}\| \leq \frac{1}{2}$ for $1 \leq j < i \leq k$ (Size reduction);

2. $\delta \|b_{i-1}\|^2 \leq \|b_i^*\|^2 + \mu_{i,i-1}^2 \|b_{i-1}^*\|^2$ for $i = 2, \ldots, k$ (Lovász condition).

Notation 2.15. We denote by $\{\tilde{b_1}, \ldots, \tilde{b_n}\}$ the LLL-reduced basis.

As the name suggests, the first condition implies length reduction. The parameter δ of the second criteria influences the goodness of the reduced basis: the higher the value is, the stronger the basis is reduced (the shorter are the basis vectors). However, this also increments the running time of the algorithm utilized to find it.

An efficient way for computing an LLL-reduced basis of a given lattice is the LLL algorithm proposed by A. Lenstra, H. Lenstra and L. Lovász. The algorithm works by iteratively transforming a given basis of the lattice into an LLL-reduced basis. The algorithm works as follows:

Algorithm 1 Lovász–Lenstra Lattice Basis Reduction Algorithm (LLL)

Require: A basis $\mathbf{B} = [b_1, b_2, \ldots, b_n]$ for a lattice $\Lambda \subset \mathbb{R}^n$, a constant $\frac{1}{4} < \delta < 1$.
Ensure: A basis $\mathbf{B}' = [b'_1, b'_2, \ldots, b'_n]$ for Λ that is LLL-reduced.

Compute the Gram-Schmidt orthogonalization $\mathbf{B}^* \leftarrow$ Gram-Schmidt($\mathbf{B}$).
Set $k \leftarrow 2$.
while $k \leq n$ **do**
    for $j \leftarrow k-1$ to 0 **do**
        $q_{k,j} \leftarrow \left\lfloor \frac{\mu_{k,j}+\frac{1}{2}}{\mu_{j,j}} \right\rfloor$, where $\mu_{i,j} = \langle \mathbf{b}_i^*, \mathbf{b}_j^* \rangle$.
        $\mathbf{b}_k \leftarrow \mathbf{b}_k - q_{k,j}\mathbf{b}_j$.
        $\mathbf{B}^* \leftarrow$ Gram-Schmidt($\mathbf{B}$).
    end for
    Compute the new Gram-Schmidt coefficients $\mu_{k,k-1}$ and $\mu_{k,k} = \|\mathbf{b}_k^*\|^2$.
    if $\delta\mu_{k,k-1}^2 > \mu_{k-1,k-1}$ **then**
        Swap $\mathbf{b}_k$ and $\mathbf{b}_{k-1}$.
        $\mathbf{B}^* \leftarrow$ Gram-Schmidt($\mathbf{B}$).
        Set $k \leftarrow \max\{k-1, 2\}$.
    else
        Set $k \leftarrow k+1$.
    end if
end while
Set $\mathbf{B}' \leftarrow \mathbf{B}$.

Example 2.16. Let's see an example of computation of an LLL-reduced basis. We take as input the vectors b_3 and b_4 of 2.3.

For convenience matters, we will now rename $b_1 = b_3 = (1,1)$ and $b_2 = b_4 = (1,2)$. Set $\delta = \frac{3}{4}$ and start by computing with Gram-Schmmidt

$$b_1^* = b_1 = (1,1)$$
$$b_2^* = b_2 - \frac{\langle b_2, b_1^* \rangle}{\langle b_1^*, b_1^* \rangle} b_1^* = (1,2) - \frac{3}{2}(1,1) = (-\frac{1}{2}, \frac{1}{2}).$$

Now, since $\mu_{2,1} = \frac{3}{2} \geq \frac{1}{2}$, the size reduction condition is not satisfied. Hence, we try to reduce its length:

$$\tilde{b_2^*} = b_2 - \lfloor \frac{3}{2} \rfloor b_1 = (1,2) - (1,1) = (0,1).$$

Check Lovász condition with length-reduced vector

$$\frac{3}{2} = \frac{3}{4} \cdot 2 \leq 1 + \frac{1}{4} \cdot 2 = \frac{3}{2}.$$

So the condition holds and therefore we have a new reduced basis given by $\tilde{b}_1 = (1,1)$ and $\tilde{b}_2 = (0,1)$.

LLL-reduced bases have some nice and useful properties.

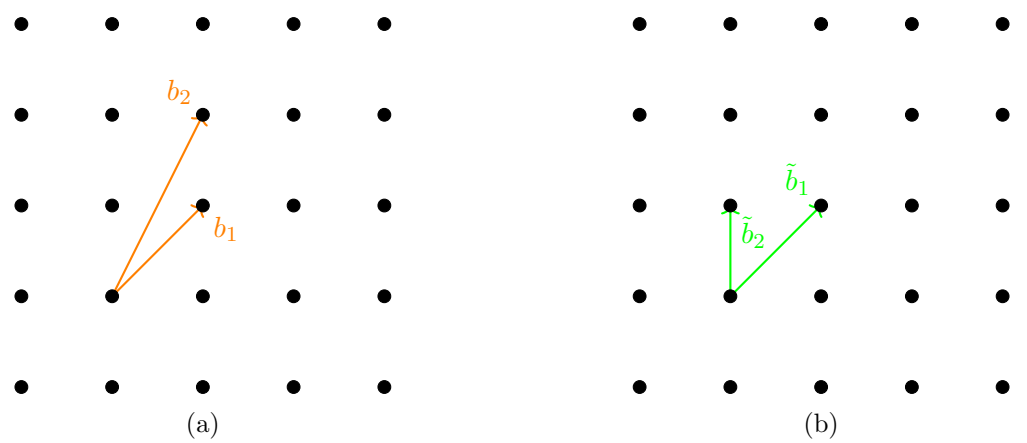

Figure 2.3: The basis vectors of a lattice $b_1 = (1, 0)$ and $b_2 = (0, 1)$ (a) and the new reduced basis after applying LLL-algorithm $\tilde{b}_1 = (1, 1)$ and $\tilde{b}_2$ (b).

Lemma 2.17. *Let $\{\tilde{b_1}, \ldots, \tilde{b_n}\}$ be an LLL-reduced basis for a lattice Λ and let $\{b_1^*, \ldots, b_n^*\}$ be the Gram-Schmidt orthogonalisation. We denote by λ_1 the length of the shortest non-zero vector in the lattice. Then*

$$\lambda_1 \geq \min_{1 \leq i \leq n} \|b_i^*\|.$$

Moreover, we define $\lambda_i = \max\{\|\tilde{b_1}\|, \ldots, \|\tilde{b_i}\|\}$.

Proof. First, we remark that every non-singular matrix B has a unique factorization

$$B = QDU,$$

where the matrices Q, D and U are orthogonal, diagonal and upper-triangular respectively. Notice that this is a QR decomposition whith $R = DU$. Now, let $B = DU$ be such factorization. Notice that we are assuming $Q = \mathbf{I_n}$. Consider then a lattice point $p = Bz$, where $z \in \mathbb{Z}^n$ is different from the null vector. Denote by z_i the last nonzero entry of z. Then, we have

$$p = Bz = DUz = D \begin{pmatrix} 1 & * & \cdots & \cdots & * \\ & 1 & \ddots & & \vdots \\ & & 1 & \ddots & \vdots \\ & 0 & & 1 & * \\ & & & & 1 \end{pmatrix} \begin{pmatrix} * \\ \vdots \\ z_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} = D \begin{pmatrix} * \\ \vdots \\ z_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} * \\ \vdots \\ \|b_i^*\| z_i \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Since $|z_i| \geq 0$, it follows that $\|p\| \geq \|b_i^*\|$ and hence the claim follows. □

Remark 2.18. Combining the above result with Minkowski's we get that the length λ_1 of the shortest non-zero vector in a lattice Λ satisfies

$$\min_{1 \leq i \leq n} \|b_i^*\| \leq \lambda_1 \leq \sqrt{n} \det(\Lambda)^{\frac{1}{n}}.$$

Theorem 2.19. *Let $\{\tilde{b_1}, \dots, \tilde{b_n}\}$ an LLL-reduced basis with $\delta = 3/4$. Then, for a lattice $\Lambda \subset \mathbb{R}^m$ generated by such basis, the following hold:*

1. $\|\tilde{b_1}\| \leq 2^{(n-1)/2}\lambda_1$;
2. $\|\tilde{b_j}\| \leq 2^{(n-1)/2}\lambda_i$; *for* $1 \leq j \leq i \leq n$
3. $2^{(1-i)/2}\lambda_i \leq \|\tilde{b_i}\| \leq 2^{(n-1)/2}\lambda_i$;
4. $\det(\Lambda) \leq \prod_{i=1}^{n} \|\tilde{b_i}\| \leq 2^{n(n-1)/4} \det(L)$;
5. $\|\tilde{b_1}\| \leq 2^{(n-1)/4} \det(\Lambda)^{1/n}$.

Proof. For additional information, we recommend consulting p. 372 of [25].

1. First, we notice that
$$\|b_j^*\|^2 \leq 2^{i-j}\|b_i^*\|^2. \tag{2.1}$$
This is due to the fact that the Lovász condition implies
$$\|b_i^*\|^2 \geq (\delta - \frac{1}{4})\|b_{i-1}^*\|^2 = \frac{1}{2}\|b_{i-1}^*\|^2.$$
Then,
$$\begin{aligned}\lambda_1 &\geq \min_{1\leq i\leq n} \|b_i^*\| \\ &\geq \min_{1\leq i\leq n} 2^{(1-i)/2}\|b_1^*\| \\ &= 2^{(1-n)/2}\|b_1^*\|.\end{aligned}$$
Since $\tilde{b_1} = b_1^*$, the result follows.

2. Let $w_1, \dots, w_i$ be linearly independent vectors in the lattice Λ s.t. $\max\{\|w_1\|, \dots, \|w_i\|\} = \lambda_i$. Then, recall that we can write each vector in the lattice as $w_j = \sum_{k=1}^{n} z_{j,k} b_k$ for $1 \leq j \leq i$ and define $k(j) = \max\{k : 1 \leq k \leq n \text{ and } z_{j,k} \neq 0\}$. It follows that
$$\|w_j\| \geq \|b_{k(j)}^*\|.$$
Now, we order the vectors w_j so that $k(1) \leq k(2) \leq \cdots \leq k(i)$. Notice also that $j \leq k(j)$. If that was not the case, then the vectors $w_1, \dots, w_j$ would be in $\text{span}\{\tilde{b}_1, \dots \tilde{b}_{j-1}\}$ which is a contradiction to their linearly independence. Then
$$\|\tilde{b_j}\| \leq 2^{(k(j)-1)/2}\|b_{k(j)}^*\| \leq 2^{(n-1)/2}\|w_j\| \leq 2^{(n-1)/2}\lambda_i,$$
which is exactly what we wanted to prove.

3. From point 2, we have the upper bound for $\|\tilde{b_i}\|$, so it only remains to show the lower one. In order to do so, we first notice that
$$\|\tilde{b_j}\| \leq 2^{(i-1)/2}\|b_i^*\| \quad \text{for} \quad 1 \leq j \leq i \leq n.$$

This is due to the fact that $j \geq 1$ implies $\frac{1}{2} + 2^{j-2} \leq 2^{j-1}$ and hence combining with eq. (2.1) we get

$$\|\tilde{b_j}\|^2 \leq 2^{i-j}2^{i-j}\|b_i^*\|^2 = 2^{i-1}\|b_i^*\|^2.$$

Taking the square root on both sides yields the result.

Now, using the fact that the vectors $\{b_1, \ldots, b_i\}$ are linearly independent, we notice that $\lambda_i \leq \max_{1\leq i\leq n} \|\tilde{b_j}\|$. Combining this with the above result we get

$$\lambda_i \leq \max_{1\leq i\leq n} \|\tilde{b_j}\| \leq 2^{(i-1)/2}\|b_i^*\|.$$

Finally, as $\|b_i^*\| \leq \|\tilde{b_i}\|$, we can conclude.

4. Since $\det(\Lambda) = \prod_{i=1}^{n} \|b_i^*\|$, and $\|b_i^*\| \leq \|\tilde{b_i}\| \leq 2^{(i-1)/2}\|b_i^*\|$, the result follows by taking the finite product on each side of the inequalities.

5. By point 4, we have $\|\tilde{b_1}\| \leq 2^{(i-1)/2}\|b_i^*\|$. Then,

$$\|\tilde{b_1}\|^n \leq \prod_{i=1}^{n} 2^{(i-1)/2}\|b_i^*\| = 2^{n(n-1)/4}\det(L).$$

Taking the n-th root on both sides yields the desired result.

□

Chapter 3

Coppersmith's Method

In 1996, Don Coppersmith published an important document [14] where he presents possible attacks on RSA cryptosystems. The main idea is to find small integer roots of a polynomial through lattice-based techniques such as LLL-algorithm. This attack allows to successfully decrypt a message when enough bits of the original message are known. Coppersmith achieved this result working on univariate polynomials modulo N and on multivariate polynomials over the integers.

3.1 Univariate modular case

In general, finding solutions to modular equations is easy if the factorization of the modulus is known. However, if that is not the case, finding solutions becomes pretty hard. For example, assume we can find a solution to $x^2 = 1 \mod N$ that is not $x = \pm 1$ then we can split N, but this is challenging when N is very large. Hence, we do not expect efficient algorithms for finding all solutions to modular equations in general. Consider a monic polynomial $F(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0$ with integer coefficients. Suppose that we know that there exists an integer solution x_0 s.t. $F(x_0) \equiv 0 \mod N$ where N is a large composite number of unknown factorization. Furthermore, assume that $|x_0| \leq N^{1/d}$. The method then proceeds by choosing a bound X s.t. $|x_0| < X$ and define the row vector associated to $F(x)$

$$b_F = (a_0, a_1 X, a_2 X^2, \ldots, a_d X^d). \tag{3.1}$$

We point out that the converse also holds: for every such row vector there is a corresponding polynomial.

Since it might be difficult to deal with polynomials whose solutions are small but not the coefficients, we want to build from $F(x)$ another polynomial $G(x)$ with relatively small coefficients and still having x_0 as a root. The following result by Howgrave-Graham will prove useful in accomplishing this task.

Theorem 3.1 (Howgrave-Graham, [26]). *Let $F(x)$ be a monic polynomial of degree d with integer coefficients and $X, N \in \mathbb{N}$. Let b_F the row vector associated to $F(x)$ defined as above. Suppose there exist some $x_0 \in \mathbb{Z}$ s.t. $|x_0| \leq X$ and $F(x_0) \equiv 0 \mod N$. If $\|b_F\| \leq N/\sqrt{d+1}$, then $F(x_0) = 0$ over the integers.*

Proof. In order to prove this result, we recall Cauchy-Schwarz inequality

$$\left(\sum_{i=1}^{n} x_i y_i\right)^2 \leq \left(\sum_{i=1}^{n} x_i^2\right)\left(\sum_{i=1}^{n} y_i^2\right).$$

Let $x_0 \geq 0$ and fix $y_i = 1$ for every $1 \leq i \leq n$. We get

$$\left(\sum_{i=1}^{n} x_i\right)^2 \leq n \sum_{i=1}^{n} x_i^2.$$

Taking the square root on both sides, the inequality becomes

$$\sum_{i=1}^{n} x_i \leq \sqrt{n \sum_{i=1}^{n} x_i^2}. \tag{3.2}$$

Now, let $F(x_0) = \sum_{i=0}^{d} a_i x_0^i$. Then

$$|F(x_0)| = \left|\sum_{i=0}^{d} a_i x_0^i\right| \leq \sum_{i=0}^{d} |a_i||x_0^i| \leq \sum_{i=0}^{d} |a_i| X^i \leq \sqrt{d+1}\|b_F\| \leq \sqrt{d+1}\frac{N}{\sqrt{d+1}} = N.$$

The second inequality is given by the fact that $|x_0| \leq X$ while the third one is given by 3.2. Since $-N < F(x_0) < N$ and $F(x_0) \equiv 0 \mod N$, we get $F(x_0) = 0$. □

As mentioned before, we have to build, starting from $F(x)$, a polynomial $G(x)$ with relatively small integer coefficients and x_0 as a root. Thanks to Theorem 3.1, the solution x_0 will also be a modular solution of $F(x)$. In order to find such $G(x)$, we define the polynomials $G_i(x) = Nx^i$ for $0 \leq i < d$. Together with $F(x)$, and associating each polynomial to a row vector, we build a $(d+1) \times (d+1)-$matrix B generating a lattice Λ:

$$B = \begin{bmatrix} N & 0 & \dots & 0 & 0 \\ 0 & NX & \dots & 0 & 0 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & NX^{d-1} & 0 \\ a_0 & a_1 X & \dots & a_{d-1}X^{d-1} & X^d \end{bmatrix}.$$

Notice that the above matrix is a lower triangular matrix. We can hence easily compute its determinant by taking the product of its diagonal

$$\det(\Lambda) = N^d \prod_{i=0}^{d} X^i = N^d X^{d(d+1)/2}.$$

We then run LLL algorithm on this matrix in order to get an LLL-reduced basis. The first row vector $\tilde{b_1}$ of the new basis $\tilde{B}$ will correspond to the polynomial $G(x)$. This can be reformulated and proved through the following theorem.

Theorem 3.2. *Let $G(x)$ be the polynomial corresponding to the first row vector of the LLL-reduced basis of a lattice $\Lambda(B)$. Set $c_1(d) = 2^{-1/2}(d+1)^{-1/d}$. If $X < c_1(d)N^{2/d(d+1)}$ then for any root x_0 of $F(x)$ modulo N s.t. $|x_0| \leq X$ we have $G(x_0) = 0$ in $\mathbb{Z}$.*

Proof. Recall from Theorem 2.19 that the first row vector of the lattice-reduced basis $\tilde{b}_1$ satisfies

$$\|\tilde{b}_1\| \leq 2^{(n-1)/4} \det(\Lambda)^{1/n}.$$

As the dimension of the lattice Λ is $d+1$, we get

$$\|\tilde{b}_1\| \leq 2^{(d+1-1)/4}(N^d X^{d(d+1)/2})^{\frac{1}{d+1}} = 2^{d/4} N^{d/(d+1)} X^{d/2}.$$

Now, we want $\tilde{b}_1$ to satisfy the Howgrave-Graham condition $\|\tilde{b}_1\| \leq N/\sqrt{d+1}$. This translates to

$$2^{d/4} N^{d/(d+1)} X^{d/2} \leq N/\sqrt{d+1}.$$

Which can be rewritten as

$$\sqrt{d+1}\, 2^{d/4} X^{d/2} \leq N^{1/(d+1)}.$$

Raising each side to the power of $\frac{2}{d}$, we get the condition of the theorem and hence we can conclude. □

We can summarize Coppersmith's algorithm as follows.

Coppersmith's Algorithm (HG version)

Given: a polynomial $F(x)$, a bound X and a modulus N
Want: a small solution x_0 such that $F(x_0) \equiv 0 \mod N$ and $|x_0| \leq X$

1. Build an upper-triangular matrix B having as diagonal elements the monomials NX^i for $0 \leq i \leq d$, as last row $F(x)$ and all remaining entries 0.

$$B = \begin{bmatrix} N & 0 & 0 & 0 & 0 \\ 0 & NX & 0 & 0 & 0 \\ 0 & 0 & NX^2 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ a_0 & a_1X & a_2X^2 & \dots & X^d \end{bmatrix}$$

2. Apply LLL-reduction on B and obtain a new basis $\tilde{B}$.
3. Take the first row vector of the reduced basis and convert it to an integer polynomial $G(x)$.
4. Apply any root finding method.
5. Verify that the solutions satisfy the bound condition.

The roots of $G(x)$ are solutions of $F(x) \equiv 0 \mod N$ as well, hence the problem is solved.

To have a clearer idea, we can examine the following example.

Example 3.3 ([26])**.** Let $N = 10001$ and

$$F(x) = x^3 + 10x^2 + 5000x - 222.$$

Suppose $X = 10$ is the bound. We want to find the small solution x_0 s.t. $F(x_0) \equiv 0 \mod N$. In order to do so, we first build the basis matrix

$$B = \begin{bmatrix} N & 0 & 0 & 0 \\ 0 & NX & 0 & 0 \\ 0 & 0 & NX^2 & 0 \\ -222 & 5000X & 10X^2 & X^3 \end{bmatrix}.$$

Then, we apply LLL-algorithm as in A.1 on B and we get

$$\tilde{B} = \begin{bmatrix} 444 & 10 & -2000 & -2000 \\ 9557 & -10 & 2000 & 2000 \\ -222 & 50000 & 1000 & 1000 \\ 989 & 2500 & 500100 & -500000 \end{bmatrix}.$$

We take the first row vector and find the corresponding polynomial

$$G(x) = 444 + x - 20x^2 - 2x^3.$$

Applying a root finding method such as Newton's, we get $x_0 = 4$. We check that it is also a root for $F(x)$:

$$F(4) = 64 + 160 + 20000 - 222 = 20002 \equiv 0 \mod N.$$

Hence we found a small solution following the method described above.

One has to be conscious that Howgrave-Graham method is a simplification of the original Coppersmith's method. In fact, the bound for X here is very small, but it can be improved so that more solutions are allowed. Analyzing the proof of Theorem 3.2, we get that the condition of success of LLL-algorithm [26] is basically

$$\det(\Lambda) < N^{d+1}. \tag{3.3}$$

We can exploit this result and develop essentially two strategies in order to achieve a larger bound:

- Increase the dimension of the matrix B by adding rows corresponding to the x-*shifts* of $F(x)$. This operation makes the determinant smaller and hence (3.3) is satisfied.
- Use powers of $F(x)$ so that the RHS of (3.3) gets larger. This is possible since $F(x_0) \equiv 0 \mod N$ implies $F(x_0)^k \equiv 0 \mod N$.

Remark 3.4. With x-shifts of $F(x)$ we refer to the polynomials $xF(x), x^2F(x), \ldots, x^kF(x)$.

We will now analyze the impact of these new strategies on the previously mentioned example.

Example 3.5. We take the same data as Example 3.3, with the distinction that in this instance the bound X must satisfy the condition (3.3). We have that the degree d of the polynomial is 3, hence we get $X \approx 2.07$.
Thus, we build this time a 6×6 matrix where the last 2 rows correspond to the shifted polynomials $xF(x)$ and $x^2F(x)$

$$B = \begin{bmatrix} N & 0 & 0 & 0 & 0 & 0 \\ 0 & NX & 0 & 0 & 0 & 0 \\ 0 & 0 & NX^2 & 0 & 0 & 0 \\ -222 & 5000X & 10X^2 & X^3 & 0 & 0 \\ 0 & -222X & 5000X^2 & 10X^3 & X^4 & 0 \\ 0 & 0 & -222X^2 & 5000X^3 & 10X^4 & X^5 \end{bmatrix}.$$

Thanks to this process, the degree becomes 5 and hence we obtain

$$2^{5/4} M^{3/6} X^{15/6} \leq \frac{N}{\sqrt{6}}.$$

Therefore, $X \approx 3.11$ which is a larger bound than before.

We can formulate the authentic result of Coppersmith in the following theorem.

Theorem 3.6 (Coppersmith)**.** *Let $F(x)$ be a univariate monic polynomial of degree d modulo an integer N whose factorization is unknown. Let $0 < \varepsilon < \min\{0.18, \frac{1}{d}\}$ and X be a bound for the solution, i.e. $|x_0| \leq X$. If $X < \frac{1}{2}N^{\frac{1}{d}-\varepsilon}$, then we can find all the integer solution x_0 s.t. $F(x_0) = 0 \mod N$ in polynomial time $(\log(N), d, 1/\varepsilon)$.*

Proof. Let $h > 1$ be an integer depending on d and ε whose value will be determined later.
Let Λ be the lattice whose rows correspond to the polynomials

$$G_{i,j}(x) = N^{h-1-j}F(x)^j x^i,$$

where $0 \leq i < d,\ 0 \leq j < h$. It follows that $\dim(\Lambda) = dh$.
Notice that if x_0 is a solution of $F(x) \equiv 0 \mod N$ then it will also be a solution of $G_{i,j}(x) \equiv 0 \mod N^{h-1}$.
By this construction, the generator matrix B of Λ ends up being a lower triangular matrix with diagonal entries $N^{h-1-j}X^{jd+i}$. Its determinant is hence the product of the diagonal

$$\det(B) = N^{(h-1)hd/2}X^{(dh-1)dh/2}.$$

We run LLL-algorithm on B and obtain the reduced basis $\tilde{B}$. The first row vector $\tilde{b_1}$ of $\tilde{B}$ satisfies

$$\|\tilde{b_1}\| \leq 2^{(dh-1)/4}\det(B)^{1/dh} = 2^{(dh-1)/4}N^{(h-1)hd/2}X^{(dh-1)dh/2}.$$

As we pointed out before, since this row vector corresponds to a polynomial $G(x)$ of degree $dh-1$, we have that $G(x_0) \equiv 0 \mod N^{h-1}$. Now, we would like to apply Howgrave-Graham's Theorem, thus we check that

$$\|\tilde{b_1}\| \leq \frac{N^{h-1}}{\sqrt{(dh-1)+1}} = N^{h-1}/\sqrt{dh}.$$

This means that we need to find the value of h such that

$$2^{(dh-1)/4}N^{(h-1)hd/2}X^{(dh-1)dh/2} \leq N^{h-1}/\sqrt{dh},$$

which is equivalent to

$$\sqrt{dh}2^{(dh-1)/4}X^{(dh-1)dh/2} \leq N^{(h-1)/2}.$$

Making some substitutions, we get

$$c_{d,h}X \leq N^{(h-1)/(dh-1)},$$

where

$$c_{d,h} = (\sqrt{dh}2^{(dh-1)/4})^{2/(dh-1)} = \sqrt{2}(dh)^{\frac{1}{(dh-1)}}.$$

We can rewrite it as

$$X \leq \frac{1}{c_{d,h}} N^{(h-1)/(dh-1)}. \tag{3.4}$$

Now, the power of N is

$$\begin{aligned}\frac{h-1}{dh-1} &= \frac{d(h-1)}{d(dh-1)} \\ &= \frac{dh-1+1-d}{d(dh-1)} \\ &= \frac{1}{d} - \frac{d-1}{d(dh-1)}.\end{aligned}$$

Set $\varepsilon = \frac{d-1}{d(dh-1)}$, $k = \frac{\varepsilon d}{d-1}$ and rewrite $dh = \frac{dh-1}{\varepsilon d} + 1$. Then, we get

$$\begin{aligned}c_{h,d} &= \sqrt{2}(dh)^{\frac{1}{(dh-1)}} \\ &= \sqrt{2}\left(1 + \frac{d-1}{\varepsilon d}\right)^{\frac{d\varepsilon}{d-1}} \\ &= \sqrt{2}\,(1+k)^{\frac{1}{k}}.\end{aligned}$$

We substitute again in eq. (3.4) and get

$$X \leq \frac{1}{\sqrt{2}\,(1+k)^{\frac{1}{k}}} N^{\frac{1}{d}-\varepsilon}.$$

We want $X \leq \frac{1}{2} N^{1/d-\varepsilon}$, hence we must have

$$\left(1 + \frac{1}{k}\right)^k \leq \sqrt{2}$$

This implies $0 \leq k \leq 0.18$. Since $k = \frac{\varepsilon d}{d-1}$, we find $\varepsilon = k(1 - \frac{1}{d}) = 0.18(1 - \frac{1}{d})$.
Then, we can set a proper bound for h. Notice that

$$h = \frac{1}{d} + \frac{d-1}{\varepsilon d^2} = \frac{1}{d\varepsilon} - \frac{1}{d^2\varepsilon} + \frac{1}{d} \approx \frac{1}{d\varepsilon}.$$

Hence, it is enough to have $h \geq \frac{1}{d\varepsilon}$ to satisfy the conditions of Howgrave-Graham result.

Finally, to see that the algorithm works in polynomial time, we notice that the dimension of the lattice is $dh \approx \frac{1}{\varepsilon}$ and that the coefficients of $G_{i,j}$ are all bounded by N^h. This means that LLL only depends on $d, \frac{1}{\varepsilon}$ and $\log(N)$.

□

Hence, we can extend the bound in exchange of some additional running time. We can still find another method to solve such problems. Let us explore this version through the following example.

Example 3.7 ([26])**.** Let $N = (20^{20}+7)(2^{21}+17)$ and $F(x) = x^3 + (2^{25} - 2883584)x^2 + 46976195x + 227$. We want to use Coppersmith's method to find an integer solution x_0 such that $|x_0| < 2^9$.

We start by building the matrix

$$B = \begin{bmatrix} N^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & N^2X & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & N^2X^2 & 0 & 0 & 0 & 0 \\ Na_0 & Na_1X & Na_2X^2 & NX^3 & 0 & 0 & 0 \\ 0 & Na_0X & Na_1X^2 & Na_2X^3 & NX^4 & 0 & 0 \\ 0 & 0 & Na_0X^2 & Na_1X^3 & Na_2X^4 & NX^5 & 0 \\ a_2^2 & 2a_0a_1X & (a_1^2 + 2a_0a_2)X^2 & (a_0^2 + 2a_1a_2)X^3 & (a_2^2 + 2a_1)X^4 & 2a_2X^5 & X^6 \end{bmatrix},$$

where the a_i's are the coefficients of x^i in $F(x)$.
We apply LLL-algorithm and obtain the LLL-reduced basis $\tilde{B}$ whose first row is

$$\tilde{b_1} = (-517019390682928109384 19, -72017716118320131768832, \dots).$$

We convert it to the corresponding polynomial of degree 6

$$\begin{aligned} G(x) = & -517019390682928109384 19 - 1406596017935940073 61x \\ & + 723517952214637262x^2 + 1311206061531618x^3 \\ & - 458777149037x^4 + 7283885822x^5 + 14268133x^6. \end{aligned}$$

Finally, we run an algorithm A.2 to find the roots of $G(x) \mod N$ and obtain the following list of solutions

$$\{267, 202361314767, 347756260719, 751773455049, 1099495374911,$$
$$1301921701650, 1447251635363, 1648907515193, 1996663775645\}.$$

Since we have the upper bound 2^9, we conclude that the only small solution to the problem is $x_0 = 267$.

3.2 Bivariate integer case

In a similar way, we can apply Coppersmith's algorithm to polynomials in two variables over the integers. Consider a polynomial $F(x, y)$ with integers coefficients and suppose that we want to find one or more solutions $(x_0, y_0) \in \mathbb{Z}^2$ such that $F(x_0, y_0) = 0$ where $|x_0| \leq X$ and $|y_0| \leq Y$. In order to solve this problem, we can use the following results.

Theorem 3.8 (Bivariate Coppersmith)**.** *Let $F(x, y)$ be a polynomial with integer coefficients whose maximum degree in each variable is d and suppose there exist at least a*

small root (x_0, y_0) *of such polynomial. Let* X, Y *be the bounds of the solutions* x_0 *and* y_0 *respectively. Define then a new polynomial* $\tilde{F}(x, y) = F(xX, yY)$ *and*

$$W = \sup\{|F_{i,j}| \ : \ F_{i,j} \text{ coefficient of } F(x, y)\}.$$

If

$$XY \leq W^{\frac{2}{3d}-\varepsilon} 2^{\frac{-14d}{3}},$$

then, there is an algorithm that can find in polynomial time $(\log W, d, 1/\varepsilon)$ *all* $(x_0, y_0) \in \mathbb{Z}^2$ *bounded by* X *and* Y *solutions of* $F(x, y)$.

Proof. The proof is quite straightforward.
We know that the generating matrix of the lattice has dimension $2kd + d^2$ with $k = \mathcal{O}(1/\varepsilon)$. Applying LLL-reduction takes polynomial time, as stated in [29]. The remaining steps of the algorithm also require polynomial time. Therefore the whole procedure is done in polynomial time $(\log W, d, 1/\varepsilon)$. □

Corollary 3.9. *Let* $F(x, y) \in \mathbb{Z}[x, y]$ *be a polynomial whose maximum degree in each variable is* d. *Write*

$$F(x, y) = \sum_{0 \leq i,j \leq d} F_{i,j} x^i y^j$$

and define

$$W = \max_{0 \leq i,j \leq d} |F_{i,j}| X^i Y^j,$$

where $X, Y \in \mathbb{N}$.
Provided that

$$XY \leq W^{2/(3d)},$$

then we can find through an algorithm all solutions $(x_0, y_0) \in \mathbb{Z}^2$ *for* $F(x_0, y_0) = 0$ *and such that* $|x_0| \leq X$ *and* $|y_0| \leq Y$ *in polynomial time* $log(W)$ *and* 2^d.

Proof. We suggest consulting the work of Coron, Kirichenko and Tibouchi [19] for a more detailed proof that the one presented by Coppersmith in [14].
□

Theorem 3.10. *Under the hypothesis of Theorem 3.8 with the addition that the total degree of* F *is* d, *the appropriate bound is given by*

$$XY \leq W^{\frac{1}{d}} 2^{\frac{-13d}{2}}.$$

Following the work of Coron in [17], we describe the algorithm as follows:

Coppersmith's Algorithm (Bivariate case)

Given: an integer polynomial $F(x, y)$, two bounds X and Y
Want: a small solution $(x_0, y_0) \in \mathbb{Z}[x, y]$ such that $F(x_0, y_0) = 0$ and $|x_0| \leq X$, $|y_0| \leq Y$.

1. Choose an appropriately large $k \in \mathbb{N}$
2. Build a matrix B having the following form

$$\begin{pmatrix} S & \star \\ M\mathbf{I_k} & 0 \\ 0 & M\mathbf{I_t} \end{pmatrix}$$

   where

   - S is a $k^2 \times k^2$ square matrix with entries $x^{i_0+i}y^{j_0+j}$ for $0 \leq i, j < d + k$ and $0 \leq i_0, j_0 < k$
   - M is the determinant of S
   - $t = (d + k)^2 - k^2$
   - $\star$ is a $k^2 \times t$ matrix where the entries are the remaining monomials $x^i y^j$ with $0 \leq i, j < d + k$

3. Compute the Hermite Normal Form of B. The resulting matrix has form

$$\begin{pmatrix} S & \star \\ 0 & T \\ 0 & 0 \end{pmatrix}$$

4. Apply LLL-reduction on T and obtain a new basis $\tilde{T}$.
5. Take the first row vector of the reduced basis and convert it to an integer polynomial $G(x, y)$.
6. Apply any root finding method.
7. Verify that the solutions satisfy the bound condition.

The roots of $G(x, y)$ are solutions of $F(x, y) = 0$ as well, hence the problem is solved.

Example 3.11 ([26])**.** Consider

$$F(x, y) = 131xy - 1400x + 20y - 1286.$$

We want to find an integer solution (x, y) such that $|x| < 30$ and $|y| < 20$. We start by

building the (13×9) matrix

$$B = \begin{bmatrix} 131X^2Y^2 & -1400X^2Y & 20XY^2 & -1286XY & 0 & 0 & 0 & 0 & 0 \\ 0 & 131X^2Y & 0 & 20XY & -1400X^2 & 0 & -1286X & 0 & 0 \\ 0 & 0 & 131XY^2 & -1400XY & 0 & 20Y^2 & 0 & -1286Y & 0 \\ 0 & 0 & 0 & 131XY & 0 & 0 & -1400X & 20Y & -1286 \\ 131^4X^2Y^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 131^4X^2Y & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & M \end{bmatrix}.$$

Then, we take the square matrix consisting of the coefficients of the first four rows and columns of B

$$S = \begin{bmatrix} a & b & c & d \\ 0 & a & 0 & c \\ 0 & 0 & a & b \\ 0 & 0 & 0 & a \end{bmatrix}.$$

As S is upper triangular, we have $\det(S) = a^4 = 131^4 = M$. Now, we can either diagonalize or find the Hermite normal form of B. In this case, we decide to run an algorithm and find the Hermite normal form.

.

$$B' = \begin{pmatrix} 131X^2Y^2 & \star & \star & \star & \star & \star & \star & \star & \star \\ 0 & 131X^2Y & \star & \star & \star & \star & \star & \star & \star \\ 0 & 0 & 131XY^2 & \star & \star & \star & \star & \star & \star \\ 0 & 0 & 0 & 131XY & \star & \star & \star & \star & \star \\ 0 & 0 & 0 & 0 & 17161X^2 & 1527329Y^2 & 1384801X & 120766935Y & 206150770 \\ 0 & 0 & 0 & 0 & 0 & 2248091Y^2 & 61264770 & 2891628500 & 87280846 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2248091X & 3282212860 & 274267102 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 294499921Y & 0 \\ 0 & 0 & 0 & 0 & & 0 & 0 & 0 & 294499921 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The pink-highlighted matrix corresponds to T. We perform LLL-reduction to the matrix T and get as first row

$$(-92669400, -68644000, 63237630, 109233040, -24514292)$$

which corresponds to the polynomial

$$\begin{aligned} G(x) &= -102966x^2 - 171610y^2 + 2107921x + 5461652y - 24514292 \\ &= -131(187132 - 16091x + 786x^2 - 41692y + 1310y^2). \end{aligned}$$

Computing resultants of $F(x, y)$ and $G(x, y)$, we find the solution

$$(x_0, y_0) = (26, 11).$$

3.3 Multivariate modular case

Some of the previously described algorithms have also been used on integer polynomials with more than two variables, as well as on bivariate modular polynomial. However, as Coppersmith himself acknowledged [14], the results are heuristic: while solutions are occasionally found, there are instances where the algorithm fails to produce results due to various reasons. While we won't delve further into this case, it is worth mentioning it to appreciate the potential of Coppersmith's ideas and acknowledge that perfecting this particular case remains an open problem. We will however see where it can find an application later on section 4.1.

3.4 Notes on Complexity

We have seen that Coppersmith's algorithm builds upon LLL. Hence, in order to study the complexity of this method, we have to consider the complexity of the LLL algorithm. Recall from Lemma 2.17 that the length of the largest nonzero vector in the lattice is defined as $\lambda_i = \max\{\|\tilde{b_1}\|, \ldots, \|\tilde{b_i}\|\}$. Now, we can introduce the following theorem to study the complexity of LLL.

Theorem 3.12 ([47], p.480)**.** *LLL algorithm accurately calculates a reduced basis of a lattice L and operates within a polynomial time complexity. The algorithm performs $\mathcal{O}(n^4 log\lambda_i)$ arithmetic operations on integers with a length of $\mathcal{O}(nlog\lambda_i)$.*

From this result it is obvious that, for a lattice with small dimension n, the algorithm works quite fast. On the other hand, when the dimension is very large, the time increases exponentially. In the case of the factorization of 2048-bit keys discussed in 5.1, the dimension of the matrix becomes very large and therefore, the required time is extremely high. In fact, researchers estimated that the corresponding converted time was around 100 CPU years [34].

Chapter 4

Applications

Coppersmith's algorithms are powerful tools which found numerous applications in various domains. In this chapter, we delve into some of them to observe where these algorithms have proven instrumental, uncovering their impact and significance in solving complex cryptographic challenges and optimizing computations. We will also see what strategies can be adopted to avoid such attacks and protect sensible data.

4.1 Stereotyped messages

Fixed padding schemes are used in cryptography to ensure that the input data to an encryption algorithm is uniformly distributed. The goal of padding is to prevent an attacker from learning information about the original plaintext by analyzing the ciphertext. In the context of RSA encryption, fixed padding schemes are used to ensure that the message being encrypted is the same length as the modulus of the public key. If the message is shorter than the modulus, it can be padded with zeros or ones to the left. If the message is longer than the modulus, it can be split into blocks and each block can be encrypted separately. In [26], Galbraith suggests, for a k-bit moduli, to take a k'-bit message and pad it using $(k - k' - 1)$ ones to its left hand side. We will adopt this scheme as the convention for fixed padding, which will be referred to as the fixed pattern scheme (FPS). we will now delve into the theoretical framework behind stereotyped messages in order to explore their functioning.

Suppose that the message m consists of two pieces: a known part $B = 2^k + \cdots + 2^l$ and an unknown part K, for example:

$$\underbrace{\text{The secret password is: }}_{B}\ \underbrace{\textit{Patichugonix}}_{K}.$$

We are also assuming that $|K| < N^{1/3}$.
Suppose we RSA-encrypt $m = B + K$ with a small exponent, say 3, so that we get

$$c = m^3 = (B + K)^3 \mod N.$$

If c is intercepted and since we know B and N, using Coppersmith's techniques we can

recover x_0 s.t.

$$F(x_0) = (B + x_0)^3 - c = 0 \mod N.$$

Application on stereotyped messages

Given: A part B of encrypted message $m = B + K$, an intercepted ciphertext $c = m^e$ and a moduli N.
Want: find the integer K to recover m.

1. Define the polynomial $F(x) = (B + x)^e - c$ where e denotes a small exponent (usually 3).
2. Write the corresponding generating matrix as in HG version of the algorithm.
3. Apply LLL-reduction to get a new basis $\tilde{B}$.
4. Take the first row vector of $\tilde{B}$ and convert it to an integer polynomial $G(x)$.
5. Apply any root finding method to get x_0.
6. Check that $F(x_0) = 0 \mod N$.

The root x_0 of $G(x)$ will correspond to K and allows to recover the original message m.

Example 4.1 ([26], pp.406-407)**.** Consider $N = 887355420159847950880463233536 1$ and suppose that Bob wants to send a 10-bit key K to Alice encrypted as

$$m = 2^{100} - 2^{10} + K.$$

Suppose further, that the ciphertext $c = 809057455777566200535445549107 6$ has been intercepted. We want to discover the value of K. Define

$$\begin{aligned} F(x) &= (2^{100} - 2^{10} + x)^3 - c \\ &= x^3 + a_2x^2 + a_1x + a_0. \end{aligned}$$

We write the corresponding generating matrix

$$B = \begin{bmatrix} N & 0 & 0 & 0 \\ 0 & NX & 0 & 0 \\ 0 & 0 & NX^2 & 0 \\ a_0 & a_1X & a_2X^2 & X^3 \end{bmatrix},$$

where $X = 2^{10}$.
As usual, we now perform LLL-algorithm on B. The first row vector of the reduced

basis corresponds to the polynomial

$$G(x) = (x-987)(-920735567540915376297+726745175435904508x+277605904865853x^2).$$

From this factorization, it is immediate that 987 is a root of $G(x)$. Finally, it is easy to check that $F(987) = 0 \mod N$.

Another interesting version where the attack works efficiently occurs when the secret part x consists of two or more blocks. Consider for example:

$$\text{"YOUR USERNAME IS } \underbrace{tichu}_{x} \text{ AND YOUR PASSWORD IS } \underbrace{oriknight}_{y}\text{"}.$$

In this case, we have that the known part is given by

$$B = \text{"YOUR USERNAME IS} \ldots \text{AND YOUR PASSWORD IS} \ldots \text{"}.$$

While the unknowns are both $x =$ "tichu" and $y =$ "oriknight".
Now, assume that we have the lengths of x and y and let

$$m = B + 2^k x + y,$$

which is encrypted as

$$c = m^3 \mod N.$$

Then, we want to find the roots (x_0, y_0) of the polynomial

$$F(x, y) = c - (B + 2^k x + y)^3 = 0 \mod N.$$

Notice that, as in the previous sections, we wish that the solutions are no larger than a certain bound. Let this bound be $3t$, where t is a positive integer such that, for the monomials $x^g y^h$, the relation $g+h < 3t$ is satisfied. The next step requires to define polynomial equations of the form

$$q_{ijk}(x, y) = x^i y^i F(x, y)^k = 0 \mod N^k.$$

where $j \leq 2, k \geq 1$ and $i + j + 3k < 3t$.
Analyzing the determinant of the corresponding matrix, we get

$$det(B) = \binom{4}{2} + \binom{7}{2} + \cdots + \binom{3t-2}{2} = \frac{3t^2(t-1)}{2}.$$

While we obtain for the powers of $1/X$ and $1/Y$

$$\binom{1}{2} + \binom{2}{2} + \cdots + \binom{3t}{2} = \binom{3t+1}{3} = \frac{27t^3 - 1}{6}$$

Combining these information. We get that the bound condition to make the algorithm work is

$$(XY)^{27t^3-1} < N^{9t^3-9t^2}$$

which is basically

$$(XY) < N^{1/3-\varepsilon},$$

where $\varepsilon \approx \frac{1}{3t}$.
Therefore, if this requirement is satisfied, we can get at least one equation $C(x, y) = 0$ and we might be able to solve the problem. We refer to pp. 254-256 of [14] for further details concerning this application.

4.2 Random Padding for Two Messages

Suppose now that we get two ciphertexts corresponding to the messages m, m' such that

$$m' = m + r,$$

where the value of r is known. Furthermore, consider that we know

$$\begin{aligned} c &= m^3 \mod N \\ c' &= (m')^3 = (m+r)^3 = m^3 + 3m^2r + 3mr^2 + r^3 \mod N. \end{aligned}$$

With such information, one can recover m (and, by consequence, also m') [14]. In fact, we can rewrite

$$m = \frac{r(c' + 2c - r^3)}{c' - c + 2r^3} = \frac{r(3m^3 + 3m^2r + 3mr^2)}{3m^2r + 3mr^2 + 3r^3} \mod N.$$

However, we want to take a further step. Assume we don't know r, hence the exact relation between m and m'. In spite of that, we know that r satisfies

$$|r| < N^{1/9}.$$

The assumption is to take messages M which have been Random Padding (RP) and then RSA-encrypted with exponent $e = 3$. Moreover, we know that a value R is added to $2^k M$ giving the plaintext $m = 2^k M + R$. Now, as before, consider

$$c = m^3 = (2^k M + R)^3 \mod N,$$

which we rewrite as

$$F = m^3 - c. \tag{4.1}$$

Suppose there is a second random pad $R' = R + r$, and define

$$m' = m + r = 2^k M + \underbrace{R + r}_{R'}.$$

Then,

$$c' = (m')^3 = (m+r)^3 \mod N$$

which becomes

$$F' = (m')^3 - c'. \tag{4.2}$$

We can now take the resultant of eqs. (4.1) and (4.2).

$$\begin{aligned} Res_m(F, F') &= r^9 + (3c - 3c')r^6 + (3c^2 + 21cc' + 3(c')^2)r^3 + (c - c')^3 \\ &= 0 \mod N. \end{aligned}$$

In this way, we remove m and get a modular polynomial of degree 9 which depends only on r. Now, if we manage to find a solution r which satisfies the bound condition, we can first apply Coppersmith's method and then Franklin and Reiter's results presented in [16, 23] in order to recover m. Finally, removing the padding from m, we get the original message M.

Remark 4.2. This attack is very effective up to 100 bits padding in the case of 1024-bit RSA keys.

Indeed, we observe the remarkable power of Coppersmith's technique once again. It highlights the ease with which a message can be recovered if the length of the RP is insufficiently large and the importance to use appropriate schemes in order to preserve information. In his original work [14], Coppersmith suggested some strategies to protect against this kind of attack:

1. **Randomization of the message:** In particular, the methods of Bellare and Rogway reported in [6] are advised. This seems to give additional randomization which is dispersed non-linearly across the message.

2. **Spread the RP into several blocks:** By distributing the random padding across multiple blocks rather than using a single contiguous block, the existing attack would need to be modified in order to be effective.

3. **Spread the RP through the whole message:** By dispersing the padding throughout the message, for instance, using two bits out of each eight-bit byte. This approach ensures that the padding is distributed across the entire message. By employing such a technique, the present attack is significantly mitigated, enhancing the security of the system.

4. **Incrementation of the amount of padding:** However, it's important to note that this may result in decreased efficiency.

5. **Encryption with larger exponents:** Apparently, the attack is effective for a RP of length up to $1/e^2$. Therefore, for 1024-bit RSA key, using $e = 7$ is already a better option.

6. **Make the RP depend deterministically on the message:** one approach is to subject the message to a hashing function and append the resulting hash value as the random padding. Consequently, two encryptions of the same message would yield identical outputs since the random padding would be the same. However, a potential weakness arises when a time-stamp is included in each message, occupying the low-order bits adjacent to the padding.

4.3 Factoring with partial knowledge

The techniques proposed by Coppersmith can also find an important application to the problem of factoring an integer in scenarios where partial information about one of its factors is already known.

Consider two prime numbers p, q and suppose that $N = pq$. Assume further, that we know a part of p, name it p_0, such that

$$p = p_0 + x_0$$

and

$$|p - p_0| < 2^k,$$

where x_0 is unknown and such that $|x_0| \leq X$. Then, we can define the polynomial $F(x) = (x + p_0)$ and claim that it has a small root modulo p. The main idea is to follow Coppersmith's usual procedure to build a lattice, find the root $\tilde{x_0}$ of the corresponding polynomial and eventually compute $\gcd(N, F(\tilde{x}))$.

We can reformulate and prove this result through the following theorem

Theorem 4.3. *Let $N = pq$ with $p < q < 2p$. Then we can find the factorization of N in polynomial time if we know the high-order $(\frac{1}{4}\log_2 N)$ bits of p, i.e. if*

$$|p - p_0| < \frac{1}{2\sqrt{2}} N^{1/4-\varepsilon}$$

for $0 < \varepsilon < 1/4$.

Proof. Let $F(x) = (x + p_0)$ and $X = \lfloor \frac{1}{2\sqrt{2}} N^{1/4-\varepsilon} \rfloor$. Then, notice that the condition $p < q < 2p$ implies

$$\sqrt{N/2} \leq p \leq \sqrt{N}.$$

Let $h \geq 4$ an integer whose value has to be determined later and define $k = 2h$. Now, take

$$N^h, N^{h-1}F(x), N^{h-2}F(x)^2, \ldots, NF(x)^{h-1}, F(x)^h, xF(x)^h, \ldots, x^{k-h}F(x)^h.$$

These $k + 1$ polynomials will build the basis B of the lattice, writing the vectors in the form of eq. (3.1). Notice that $\dim(B) = k + 1$ and $\det(B) = N^{h(h+1)/2} X^{k(k+1)/2}$. We want to apply LLL-reduction together with Howgrave-Graham's result, which means that we must satisfy the condition

$$2^{k/4} \det(B)^{1/(k+1)} < p^h / \sqrt{k+1}.$$

Since $\sqrt{N/2} \leq p$, the above inequality restricts to showing

$$\sqrt{k+1} \cdot 2^{k/4} \det(B)^{1/(k+1)} < (\sqrt{N/2})^h$$

which becomes

$$\sqrt{k+1} \cdot 2^{k/4} N^{h(h+1)/2(k+1)} X^{k/2} < (N/2)^{h/2}.$$

Now, we isolate the bound X

$$X < N^{h/k - h(h+1)/(k(k+1))} 2^{-h/k} \frac{2^{-1/2}}{(k+1)^{1/k}}.$$

Since $h \geq 4$, $k \geq 8$ and hence, $(k+1)^{1/k} = 2^{\log_2(k+1)/k} \leq \sqrt{2}$. From this follows

$$\frac{1}{(k+1)^{1/k}} \geq 1/\sqrt{2}.$$

Using $k = 2h$, we find that the condition to satisfy is

$$X < \frac{1}{2\sqrt{2}} N^{1/2(1-(h+1)/(2h+1))}.$$

Finally, since

$$1/2(1 - (h+1)/(2h+1)) = \frac{1}{4} - \frac{1}{(4(2h+1))},$$

we conclude that the result holds if

$$\frac{1}{(4(2h+1))} < \varepsilon.$$

In order for this to be true, it is enough to set $h \geq \max\{4, 1/(4\varepsilon)\}$.

□

Application on partial knowledge

Given: a modulo N, a bound X and p_0 s.t. $p = p_0 + x_0$ and $|p - p_0| < \frac{1}{2\sqrt{2}} N^{1/4-\varepsilon}$

Want: find the factors p, q of N.

1. Define a polynomial $F(x) = F(x + p_0)$.
2. Choose an appropriate $k \in \mathbb{N}$.
3. Build an upper-triangular matrix B having as row vectors the polynomials $F(x)$ and its $x-$shifts up to k. All remaining entries 0.
4. Apply LLL-reduction on B and obtain a new basis $\tilde{B}$.
5. Take the first row vector of the reduced basis and convert it to an integer polynomial $G(x)$.
6. Apply any root finding method and find x_0.
7. Compute $F(x_0)$.
8. Compute $\gcd(N, F(x_0))$.

The factors of N are primes, hence $F(x_0) = p$ and $\gcd(N, F(x_0)) = q$.

Example 4.4 ([26])**.** Let $N = 22461580086470571723189523$ and $p_0 = 2736273600000$ which is a correct approximation of p up to $0 \leq x < 50000$. We want to apply Coppersmith's method to find the factorization of N.

We start by defining the polynomial $F(x) = (x + p_0)$. Then we compute its $x-$shifts

$$xF(x) = (x^2 + p_0 x)$$
$$x^2 F(x) = (x^3 + p_0 x^2).$$

We build then the corresponding lattice

$$B = \begin{pmatrix} N & 0 & 0 & 0 \\ p_0 & X & 0 & 0 \\ 0 & p_0 X & X^2 & 0 \\ 0 & 0 & p_0 X^2 & X^3 \end{pmatrix}.$$

The first row of the reduced basis is

$$(2196789523, -8118700000, 7500000000),$$

which coincides with the polynomial

$$G(x) = 3x^2 - 162374x + 2196789523.$$

We find that its only integer root is 27389, therefore $p = F(27389) = 2736273627389$ and $q = \gcd(N, p) = 8208820880207$. Therefore, we were able to find the factorization of N knowing only partial information about p.

It is interesting to notice that this application can also be found in bivariate polynomials. The major changes are given by the fact that we have to choose another bound Y for y_0 and write $q = q_0 + y_0$. Then, one should define the polynomial $F(x, y) = (p_0 + x_0)(q_0 + y_0)$ and proceed analogously.

4.4 Factoring $p^r q$

Sometimes, it is convenient to use moduli of form $N = p^r q$ for two primes p, q and $r \in \mathbb{N}$. They allow a faster RSA decryption and are, for instance, commonly found in electronic cash schemes [24]. However, it is advised to use such moduli cautiously as there are many potential risks. In fact, if, as before, a part p_0 of p is known, one can use the same strategy exposed in section 4.3 to find a small integer solution of

$$F(x) = (p_0 + x)^r \equiv 0 \mod p^r.$$

The algorithm is pretty much the same as the one presented in section 4.3.

Application on partial knowledge with exponent

Given: a modulo of form N, a bound X, an integer r and p_0 s.t. $p = p_0 + x_0$ and $|p - p_0| < \frac{1}{2\sqrt{2}} N^{1/4-\varepsilon}$

Want: find the factors p, q of N s.t. $N = p^r q$.

1. Define a polynomial $F(x) = F(x + p_0)^r$.
2. Choose an appropriate $k \in \mathbb{N}$.
3. Build an upper-triangular matrix B having as row vectors the polynomials $F(x)$ and its $x-$shifts up to k. All remaining entries 0.
4. Apply LLL-reduction on B and obtain a new basis $\tilde{B}$.
5. Take the first row vector of the reduced basis and convert it to an integer polynomial $G(x)$.
6. Apply any root finding method and find x_0.
7. Compute $F(x_0)$.
8. Compute $\gcd(N, F(x_0))$.

The factors of N are both primes hence $F(x_0) = p$ and $\gcd(N, F(x_0)^r) = q$.

As usual, we work on an example to understand better the case.

Example 4.5 ([26])**.** We know that $N = 876701170324027 = p^3 q$ for some primes p, q. Moreover, $|p - 5000| < 10$. We want to find the factors p and q using Coppersmith's technique.

Same as before, we start by writing

$$F(x) = (x + p_0)^3 = x^3 + 15000x^2 + 125000000000x + 75000000.$$

Then, let $a = 125000000000, b = 75000000, c = 15000$ and compute the $x-$shifts

$$xF(x) = x(x + p_0)^3 = x^4 + cx^3 + bx^2 + ax$$
$$x^2F(x) = x^2(x + p_0)^3 = x^5 + cx^4 + bx^3 + ax^2.$$

The corresponding lattice is

$$B = \begin{pmatrix} N & 0 & 0 & 0 & 0 & 0 \\ a & bX & cX^2 & X^3 & 0 & 0 \\ 0 & aX & bX^2 & cX^3 & X^4 & 0 \\ 0 & 0 & aX^2 & bX^3 & cX^4 & X^5 \end{pmatrix}.$$

After performing LLL-reduction with A.5, the first row vector of the reduced basis is

$$(-16661499957, 32250000000, 65064500000, 41840043000, 125130000, 100000),$$

which coincides with the polynomial

$$\begin{aligned} G(x) = &-16661499957 + 3225000000x + 650645000x^2 \\ &+ 41840043x^3 + 12513x^4 + x^5. \end{aligned}$$

We find that its only integer root is 3, therefore $p = F(3) = 5003$ and $q = \gcd(N, p^3) = 7001$. Hence $N = 5003^3 \cdot 7001$. Therefore, we managed once again to find the factorization of N knowing only partial information about p.

An interesting fact is that, usually, one uses Elliptic Curves Methods (ECM) to find the factorization of such N. However, this method can be faster in many instances. For example, it has been proved that for $r = \log(p)$ it is better to use this algorithm since it is asymptotically faster than ECM. [10]

4.5 Chinese Remaindering with Errors

We recall one of the most remarkable theorems of algebra

Theorem 4.6 (Chinese Remainder Theorem (CRT))**.** *Let $n_1, n_2, \ldots, n_k$ be relatively prime positive integers and $r_1, \ldots, r_k$ integers such that $r_i \in \mathbb{Z}_{n_i}$. Then the system of simultaneous congruences*

$$\begin{aligned} x &\equiv r_1 \mod n_1 \\ x &\equiv r_2 \mod n_2 \\ &\vdots \\ x &\equiv r_n \mod n_k \end{aligned}$$

has only one solution modulo $N = n_1 n_2 \ldots n_k$.

Chinese Remainder Codes utilize the Chinese Remainder Theorem to encode and decode information. They are a class of error-correcting codes that allow efficient encoding and decoding of messages by exploiting the properties of the Chinese Remainder Theorem.

Definition 4.7 (CRT list decoding problem)**.** Let $n_1, n_2, \ldots, n_k$ and $r_1, \ldots, r_k$ be positive integers such that $p_i < p_j$ and $r_i < p_i$ for all $1 \leq i < j \leq k$. Let further X and $e \leq k$ be integers. We want to compute $x \in \mathbb{Z},\ 0 \leq x < X$ such that

$$x \equiv r_i \mod n_i$$

for every $1 \leq i < k,\ i \neq e$. This problem is named Chinese remaindering with errors problem or CRT List Decoding Problem (CRTLDP).

Remark 4.8. The problem may lack a solution depending on the values of X and e.

Drawing upon the groundwork laid by Goldreich, Ron, and Sudan [27], Boneh utilized concepts highly similar to Coppersmith's approach to devise an algorithm [9] for the following problem. The algorithm is outlined below.

Application on CRT problems (Boneh algorithm)

Given: all instances of the Chinese remaindering with errors.

Want: construct a polynomial $F(x)$ whose roots $|x| \leq X$ are solutions of the CRT list decoding problem.

1. Define $P = \prod_{i=1}^{n} p_i$.
2. Define the amplitude $\text{amp}(x) = \gcd(P, x - R)$, where R denotes the solution of the system of congruences.
3. Write $F(x) = x - R$. We look for $|x_0| \leq X$ such that $F(x_0) \equiv 0 \mod M$ for some large $M|P$.
4. Build the corresponding lattice.
5. Apply HG algorithm to find a suitable root.

The solutions of $F(x)$ are solutions of the CRT list decoding problem as well.

We recall the definition of smoothness for integers.

Definition 4.9. An integer t is $B-$smooth if t has no divisor greater than B.

Coppersmith's ideas in the formulation of Boneh's CRT list decoding algorithm, also find an application in identifying $B-$smooth integers. In fact, it is possible to use it to find $x \in \mathbb{Z}$ $B-$smooth in relatively short intervals quite efficiently. The following theorem and its proof provide insights for the formulation of such algorithm.

Theorem 4.10. *Let B, e, $p_1, \ldots, p_n$ and $r_1, \ldots, r_n$ be an instance of the list decoding problem and define $P = \prod_{i=1}^{n} p_i$. If*

$$e \leq n - n\frac{\log(p)}{\log(p_1)}\sqrt{\log(X)/\log(P)}$$

then, there exist an algorithm which finds in polynomial time all integers x satisfying $x < B$ and $x \equiv r_i \mod p_i$ for all $1 \leq i \leq n$ different from e.

Proof. The original proof of this result is presented by Boneh in [9]. However, we will follow the work of Galbraith [26] in turn based on May's [31].

Let $0 \leq x < X$ and define $M = \text{amp}(x)$. Suppose that x is divisible by at least $n-e$ values of p_i. Then, since

$$n \log(p_1) \leq \log(P) \leq n \log(p_n)$$

we also have

$$(n-e)\log(p_1) \leq M \leq n \log(p_n).$$

Rewrite $M = P^\beta$ and notice that the condition for Coppersmith's algorithm to work is essentially $X < P^{\beta^2}$. Hence, we want to find a bound for e so that

$$\frac{\log(X)}{\log(P)} < \beta^2$$

which is equivalent to

$$\sqrt{\frac{\log(X)}{\log(P)}} > \beta.$$

Since $\beta = \log(M)/\log(P)$, we get

$$\beta \geq (n-e)\log(p_1)/(n\log(p_n)).$$

Therefore, it is sufficient that

$$(n-e)\frac{\log(p_1)}{\log(p_n)} \geq n\sqrt{\frac{\log(X)}{\log(P)}}$$

Rearranging the terms, we get

$$e \leq n - n\frac{\log(p)}{\log(p_1)}\sqrt{\log(X)/\log(P)}$$

which is exactly what we wanted to get. □

Remark 4.11. To build the basis of the lattice, one must compute the polynomials

$$G_i(x) = P^{a-i}(x-R), \quad \text{for } 0 \leq i < a$$
$$H_i(x) = (x-R)^a x^i, \quad \text{for } 0 \leq i < \tilde{a}$$

for an appropriate choice of a and $\tilde{a}$.

We can formulate the algorithm for this case as follows.

Application on smooth integers

Given: An interval $I = [a, b] = \{x \in \mathbb{Z} : a \leq x \leq b\}$, $P = \prod_{i=1}^{n} p_i^{e_i}$
Want: find an integer x divisible by all $p_i \leq B$, $B \in \mathbb{N}$ (i.e. x is $B-$smooth).

1. Assume $b < 2a$ and define $W = \frac{(a+b)}{2}$, $X = \frac{b-a}{2}$
2. Compute the polynomials

$$G_i(x) = P^{a-i}(x - R) \quad \text{for } 0 \leq i < a$$
$$H_i(x) = (x - R)^a x^i \quad \text{for } 0 \leq i < \tilde{a}$$

   for an appropriate choice of a and $\tilde{a}$ and build the corresponding lattice.
3. Apply LLL-reduction.
4. Convert the first row of the reduced basis to a polynomial $F(x)$ and find its roots.
5. Verify that the factorization of $F(x) = x - R$ contains most of the factors of P.

The solution x is $B-$smooth as requested.

Example 4.12 ([26])**.** Let $P = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = 232792560$, $W = 150000001 = 1.5 \cdot 10^8 + 1$ and $X = 46558000$. We want to find an integer $x \in [W - X, W + X] = I$ such that most of the factors of P divide x.

We choose $R = -W, a = 4, \tilde{a} = 3$ and using the polynomials defined in 4.11, we build the corresponding lattice

$$B = \begin{bmatrix} P^4 & 0 & 0 & 0 & 0 & 0 & 0 \\ -RP^3 & P^3X & 0 & 0 & 0 & 0 & 0 \\ R^2P^2 & -2RP^2X & P^2X^2 & 0 & 0 & 0 & 0 \\ -R^3P & 3R^2PX & -3RPX^2 & PX^3 & 0 & 0 & 0 \\ R^4 & -4R^3X & 6R^2X^2 & -4RX^3 & X^4 & 0 & 0 \\ 0 & R^4X & -4R^3X^2 & 6R^2X^3 & -4RX^4 & X^5 & 0 \\ 0 & 0 & R^4X^2 & -4R^3X^3 & 6R^2X^4 & -4R^2X^5 & X^6 \end{bmatrix}.$$

The first row vector of the LLL-reduced basis corresponds to the polynomial

$$F(x) = 2(-82792559 + x)^3(33603721 + x)$$

which has solution $x_1 = -33603721$ and $x_2 = 82792559$. Since x_1 doesn't belong to the interval I, we don't take it in consideration. Hence, we only have to compute

$$W - x_2 = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = P,$$

which is obviously $P-$smooth. Therefore, we found an integer solution satisfying the requirements.

Remark 4.13. The probability of success of this method is still to determine.

Chapter 5

The ROCA Vulnerability

In 2017, a team of researchers from Masaryk University in the Czech Republic, discovered a dangerous vulnerability in the implementation of the RSA Key generation algorithm. This vulnerability, named the Return of Coppersmith's Attack (ROCA), takes advantage of Coppersmith's method to factor the RSA modulus, compromising the security of the affected systems. It was also added to the Common Vulnerabilities and Exposures (CVE) database under the identifier CVE-2017-15361 in order to facilitate analysis for the cybersecurity community and spread awareness.

5.1 Technical details and Key generation process

The core weakness behind the ROCA vulnerability arises from the flawed generation of prime numbers used in a software library - *RSALib* - developed by Infineon Technologies, which is widely incorporated into smart cards, Trusted Platform Modules (TPM), and Hardware Security Modules (HSM), including popular devices like YubiKey 4 tokens used for Pretty Good Privacy (PGP) key generation. When weak random number generators or faulty libraries are employed, certain primes exhibit a property called 'non-uniformity'. This property enables attackers to exploit the vulnerability and retrieve the private key corresponding to a given public key, undermining the security guaranteed by RSA-based cryptographic systems. In order to speed-up the process, the primes in the RSALib had the following structure:

$$p = (k * M + (65537^a \mod M))$$

where a, k are unknown integers, while M is known and is a primorial number, meaning the product of all primes up to n: $M = \prod_{i=1}^{n} p_i$. The reason behind this structure relies on the fact that 65537 is invertible mod M. In this way, the manufacturers had to check only the bigger factors in order to have p prime, since they knew that all small primes weren't factors of p. Furthermore, the value of M depends on the size of the key to generate. We know for example that for an RSA key with size between 512 and 960 bits, $n = 39$. This means that $M = 2 * 3 * \cdots * 39 = 7420738134810$ [34]. Other known values are represented in the following table. One can notice how the size of M is rather large compared to the size of primes. This constitutes an issue because in this

way, we can deduce that the size of a and k is small. In fact, if we consider again the example for $n = 39$, we find that M has around 219 bits and hence the size (in bits) of k is

$$256 - 219 = 37$$

Key Size (bit)	n	M
512-960	39	$P_{39}\#$
992-1952	71	$P_{71}\#$
1984-3936	126	$P_{126}\#$
3968-4096	225	$P_{225}\#$

Table 5.1: Different key sizes and their respective primorial.

Then, we can compute an upper bound for a by solving

$$65537^k \equiv 1 \mod M.$$

We find easily that a has 62 bits. This means that the resulting constructed prime p (always assuming that the RSA-key has size 512 bits) has an entropy of $37 + 62 = 99$ bits and therefore, the possible choices for the prime generated are reduced from 2^{256} to 2^{99}. This is still a huge number to use a brute-force attack, but we can already see that the generation process is definitely risky.

Now, what happens to the RSA modulo? The RSA modulus generated by such primes is

$$N = \underbrace{(k * M + (65537^a \mod M))}_{p} * \underbrace{(l * M + (65537^b \mod M))}_{q}.$$

However, notice that for $g = 65537$

$$\begin{aligned} N &= p * q \\ &= (k * M + (g^a \mod M)) * (l * M + (g^b \mod M)) \\ &= klM^2 + kM(g^b \mod M) + lM(g^a \mod M) + (g^a g^b \mod M) \\ &= (klM + ((kg^a + lg^b) \mod M)) * M + (g^{a+b} \mod M) \end{aligned}.$$

Defining $\kappa = (klM + ((kg^a + lg^b) \mod M))$ and $c = a + b$, we get

$$N = \kappa * M + (g^c \mod M)$$

which has exactly the same structure of the primes generated by RSALib. Therefore, even in this case we can restrict the choices. In fact, gathering and analyzing different public keys, researchers [22, 34] made the following discoveries:

$$\begin{aligned} N &\mod 11 \in \{1, 10\} \\ N &\mod 13 \in \{1, 3, 4, 9, 10, 12\} \\ N &\mod 17 \in \{1, 2, 4, 8, 9, 13, 15, 16\} \\ N &\mod 19 \in \{1, 4, 5, 6, 7, 9, 11, 16, 17\} \\ N &\mod 37 \in \{1, 10, 26\} \\ &\vdots \end{aligned}$$

which means that the choices for the remainders of N are confined to these small subsets and therefore, easier to find. If the prime factors were picked randomly, this wouldn't be the case; we would have had $p - 1$ possibilities for a remainder modulo p.

Another problematic characteristic that such generated N possesses is the following:

$$N \equiv 65537^c \mod M.$$

We analyze now why these findings are dangerous.
Take again $g = 65537$. We can find the order of g modulo p and use such information to compute the order of the group $G := (\mathbb{Z}/M\mathbb{Z})^* \cong (\mathbb{Z}/p_1\mathbb{Z})^* \times \cdots \times (\mathbb{Z}/p_n\mathbb{Z})^*$. This allows then to determine all other elements in it. In fact, by using Pohlig-Hellman algorithm described below, one can solve quite easily the Discrete Logarithm Problem (DLP), which means finding $\beta \in G$ (if it exists) such that

$$N \mod (M) = g^\beta.$$

Pohlig-Hellman Algorithm

Given: A generator g of a cyclic group G with $ord(g) = n = \prod_{i=1}^{r} p_i^{e_i}$ and an element $h \in G$.
Want: Find an integer $x \in \{0, \ldots, n-1\}$ such that $g^x = h$.

1. For $i \in \{0, \ldots, r\}$
   (a) Compute $g_i = g^{n/p_i^{e_i}}$ $\leftarrow$ By Lagrange theorem, $ord(g_i) = p_i^{e_i}$;
   (b) Compute $h_i = h^{n/p_i^{e_i}}$ $\leftarrow$ By construction, $h_i \in \langle g_i \rangle$;
   (c) Compute $x_i \in \{0, \ldots, p_i^{e_i} - 1\}$ such that $g_i^{x_i} = h_i$.
2. Solve the simultaneouss congruences

$$x_i \equiv x \mod p_i^{e_i}.$$

The CRT guaranties a unique solution x.

The DLP is in general very hard to solve, but in this case, the fact that M is a primorial number and, by consequence, also smooth implies that the size of the multiplicative group $\mathbb{Z}_M^*$ is again smooth and corresponds to $\varphi(M)$. This allows to efficiently compute the solution through the algorithm.

For example, in the case of a 512-bit key, we have that the size of the multiplicative group $\mathbb{Z}_M^*$ is $2^{215.98} = \varphi(M)$. On the other hand, the size of the subgroup G generated by $g = 65537$ of $\mathbb{Z}_M^*$ is $2^{61.09}$ [22]. With such information, one can compute the probability that a random 512-bit modulus belongs to G:

$$2^{62} - 2^{216} = 2^{-154}$$

which is a very low probability. This emphasize why choosing a modulus randomly would be a better option than using the described generation process. We now examine how Coppersmith's results were applied to find the factorization of such keys.

We recall that the main condition for the HG version of Coppersmith's algorithm to work is that at least $\log_2(N)/4$ bits are known.

We have

$$p = (k * M + (65537^a \mod M))$$

Together with the fact that k is small and

$$\log_2(M) > \log_2(N)/4,$$

we can adapt the method to factorize N using $p \equiv 65537^a \mod M$.

In order to do so, we first define the polynomial

$$F(x) = x * M + (65537^a \mod M) \mod N.$$

We want to ensure that the polynomial is monic, so we perform some computations:

$$\begin{aligned} F(x) &= x * M * (M^{-1} \mod N) + (65537^a \mod M)) * (M^{-1} \mod N) \\ &= x + (65537^a \mod M) * (M^{-1} \mod N) \mod N. \end{aligned}$$

Now, one has the following parameters to set:

- $f(x)$: The above defined polynomial which will build the lattice.
- N: The modulus we want to factorize.
- X: Upper bound for the small solution x_0.
- β: Lower bound for the prime p such that $p|N$ and $p \geq N^\beta$. Moreover, $0 < \beta < 1$.

Once that these parameters are settled, one can apply Coppersmith's algorithm in the appropriate form such as the HG version presented in Section 3.1.

5.1.1 Simulations

We tried to perform some simulations to have a wider idea of the power and efficiency of the attack. In order to do so, we used the code proposed by ENOENT in [22]. We managed to factorize efficiently some 512-bit keys. However, we could not represent successfully the data.

5.2 ROCA in Estonia

As the time advances, society tries to evolve accordingly. One example can be observed in the substitution of traditional physical documents such as identity cards, passports, driving licenses, to electronic copies. Estonia is recognized to be one of the first countries implementing an Electronic Identification (eID) program back in 2002 [36]. This innovation was meant to simplify the use of different government services by making them accessible online: giving digital signatures the same value as handwritten ones allows people to authenticate themselves and perform everyday operations anywhere and anytime.

On August 2017, the Estonian government was notified by a research group from Masaryk University [34] about a possible vulnerability found in the software embedded into their eIDs. According to the researchers, the keypairs used in the chips produced by Infineon Technologies, were generated by using the flawed process presented in Section 5.1 which allows third parties to exploit Coppersmith's method and give birth to the ROCA vulnerability. Analysis revealed that the compromised eIDs had been in circulation since 2014 and it was subsequently estimated that a minimum of 1 billion of chips exploited in different mechanism - from computing devices to plastic cards - were affected around the world [46]. It is known that in Estonia alone, approximately 750,000 eID's - more than half the population's - were flawed [36, 37]. Although, none of them was reportedly compromised, the gravity of this situation cannot be overstated. In fact, a successful exploitation of the ROCA vulnerability could have empowered attackers with the capability to decrypt sensitive documents or potentially clone them without requiring access to a physical copy of the documents. Moreover, this could have been achieved with considerably low computing power and costs than what it should have been. Experts estabilished that such costs were around or lower than USD 25,000 for 2048 bit RSA keys, when at first it was estimated at least USD 80,000 [45].

It was clear that there was a flaw that had to be fixed in order to minimize the risks. However, this had to be done while maintaining an operating e-State [37]. How did the Estonian government deal with this situation? The authorities decided to be transparent and disclose such information to its citizens before the paper concerning the details of the attack was released. In this way, they could build a sense of trust between people. At the same time, they could focus on fixing the problem. The solution was presented a couple of months later, on October 25$^{\text{th}}$, when it was decided to replace the RSA library with Elliptic Curves Cryptography (ECC) encryption algorithms. This allowed cardholders to efficiently patch their eIDs remotely by using a software provided by the State which anyone could use on any computer. It is estimated that in less than a week, 20'000 affected eID cards were updated. A few days later, on October 30$^{\text{th}}$, the paper

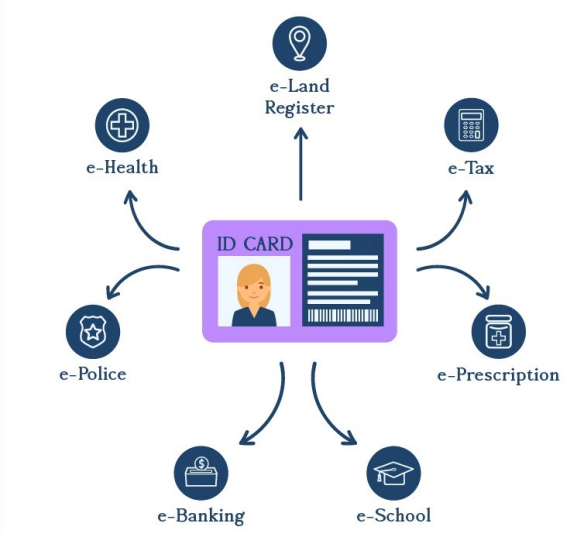

Figure 5.1: Examples of eIDs uses in Estonia.

containing the details of the attack and vulnerability [34] was published. Within days, it was recorded that 740'000 flawed Estonian eID's were blocked. They were however reclaimable if updated online with the software. To facilitate this operation, the Police and Border Guard (PPA), opened several offices to help anyone updating the certificates. By the end of 2017, 400'000 eIDs - more than a half of the affected ones - were updated and serviceable. The renovation process and possibility to update certificates went on until the end of March 2018. From this point on, any non-updated certificate was revoked and no longer usable.

5.3 Prevention against future Attacks

The response and strategy proposed by Estonian against the ROCA vulnerability can be considered quite a success since, in the end, 94% of eIDs were updated and none of them was reportedly compromised [3]. It is also interesting to notice that in October 2017 local elections were held and, despite the situation, the percentage of i-Voters didn't decrease. Conversely, it was the highest ever registered up to that time (31,7% out of the total voters were i-Voters) [1, 3]. Furthermore, the amount of transactions involving eIDs was not affected. This seems to suggest that, although the crisis, confidence in the e-State was maintained. In support of this hypothesis we can compare the following data: it is recorded that, in February 2017, 6 million digital signatures were given, whereas in the same month a year later, the number had risen to 10 million.

Despite this efficiency, one major question arises. How can one be better prepared for the next vulnerability?

The first improvement suggested concerns the notification system. The notification

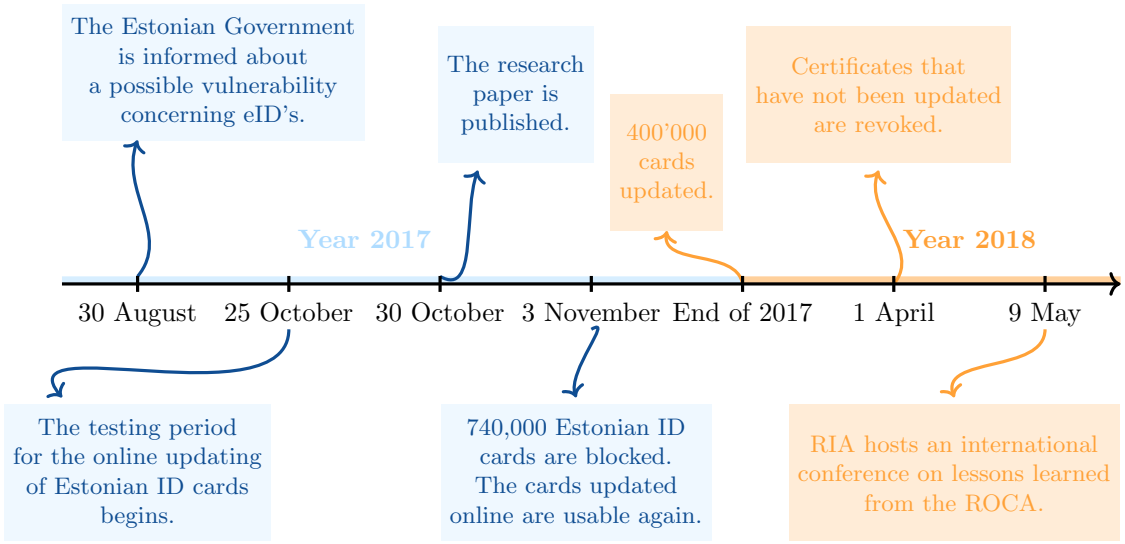

Figure 5.2: Timeline of the discovery of the ROCA vulnerability in Estonian eIDs and the adopted countermeasures.

mechanism in Estonia, probably failed this situation because it was designed to manage situations that have already caused visible effects, and hence they might not be well-equipped to deal with vulnerabilities in the initial phases of a crisis [3]. So, instead of relying fully on a traditional notification mechanism, it could be a good idea to collaborate with international groups of researchers and entities. Indeed, this would provide alternative sources of information and a higher probability to avoid similar crisis.

An additional matter that requires attention is developing an alternative to the eIDs. As already discussed above, eIDs are indispensable to access a wide array of both public and private services in Estonia. In addition, it is important to note that other options such as in-person authentication or physical handwritten signatures are not considered viable. [3] This means that the state depends completely on eIDs and therefore, when a vulnerability arises, it poses a significant and evident threat to the state's security or stability. To mitigate this risk, it could be advisable to add an alternative method to access to the essential services even in a worst-case scenario where eIDs become compromised or unavailable.

Another crucial aspect that must be considered concerns the manufacturers. Using shortcuts to generate prime numbers may seem convenient, but it can undermine the fundamental nature of prime numbers chosen at random and the security of cryptographic systems that rely on them. Taking shortcuts can lead to predictable patterns, such as the one presented in Section 5.1, which can then give rise to vulnerabilities. In contrast, selecting prime numbers randomly enhances security because it introduces unpredictability and makes it significantly more difficult for adversaries to decipher encrypted data or break cryptographic protocols. Therefore, it is important to do rigor-

ous research and tests before contemplating the implementation of alternative methods, such as shortcuts for prime generation, especially when the target is distributed on such a large-scale.

5.4 Other global impacts

As Estonia relies a lot on e-services, the ROCA incident in the country stands out as particularly remarkable due to its significant impact. Nevertheless, it is worth mentioning a few other affected parties.

As already pointed out above, it was estimated that over 1 billion chips used worldwide were vulnerable. Among the known affected countries, we find Slovakia, Spain, Taiwan, Poland, Italy, Brazil, Austria, Bulgaria, Kosovo and Malaysia. It is very difficult to find information about how the situation was managed in these states and make a comparison with Estonia. This is probably due to the fact that the mentioned countries depended far less than the latter on e-services and thus may not have had the same urgency to respond quickly. However, we were able to uncover some specific data: in Spain alone at least 17 million cards (representing $31,5\%$ of the whole population at that time) were compromised and revoked. In Slovakia, this was done for $60,000$ certificates (1,1% of the whole population) [3]. Notice however, that due to the lack of data, we could not find the actual number of e-card holders, hence the percentages are based on the whole population and therefore cannot be considered much accurate.

The ROCA vulnerability also impacted TPMs. TPMs are at the basis of modern computers' security and are mainly employed for authentication of hardware and software. There are two RSA keys in TPMs: the Endorsement Key (EK) which is a unique key associated with the TPM during its manufacturing process used for attestation and endorsement of the TPM's identity, and the Storage Root Key (SRK), which is generated after taking ownership of the TPM and is used to protect keys that may reside outside the TPM itself. Usually, TPMs are embedded inside the device in form of a chip therefore, are hard to replace [2]. Keys that have been enveloped by the SRK can in turn be employed to envelop additional keys, forming a hierarchical structure of parent and child keys. To load a child key, it's necessary to load its parent key first. Once the child key is successfully loaded, the parent key can be unloaded from the TPM, thereby freeing up valuable TPM chip resources. This hierarchical approach enables efficient and secure key management within the TPM system. The problem which arises by factoring EK represents a critical compromise to the foundation of trust for chip authentication. An attacker can create a new keypair external to the TPM and then use the factorized EK to sign it. Consequently, this newly generated keypair would gain the trust of the remote system, such as a corporate network, as it appears to be endorsed by the compromised EK. This security breach can have significant implications for the integrity of the system and its authentication mechanisms. Several technological companies and corporations including Microsoft, Google, Lenovo, HP, Toshiba and Fujitsu recognized to have devices affected by the vulnerability. [34] In order to protect users,

they released security reports and proposed patches which temporarily replaced TPM. Moreover, individuals could check whether their keys were affected by using the detection tools[1] published by the discoverers of the vulnerability [48].

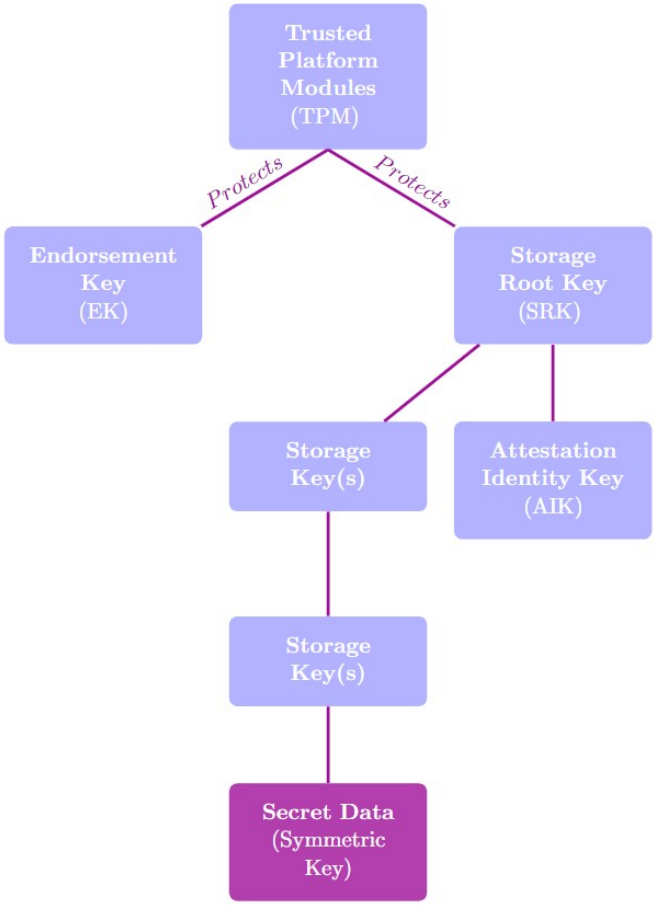

Figure 5.3: Hierarchy of TPMs.

Another widely affected domain were PGP, a data encryption and decryption program used for securing e-mails, files, digital signatures and other forms of digital communication. If compromised, an attacker gains the ability to forge new signatures and decrypt all previously intercepted messages, as PGP does not offer forward secrecy [44]. A significant portion of public keys used with PGP can be easily retrieved from PGP key servers. As the content of PGP key servers is publicly accessible, it becomes straightforward to identify vulnerable keys along with their associated user contact information. For the researchers, it was then possible to conduct some analysis. It emerged that one of the most affected source in the PGP dataset was Yubikey 4, a hardware authentication device manifactured by Yubico [34]. Its primary purpose is to enhance security by

[1] `https://github.com/nsacyber/Detect-CVE-2017-15361-TPM`

safeguarding access to computers, networks, and online services. Yubico addressed this issue by implementing a change in the key generation function for all YubiKey 4 devices that were being shipped. Additionally, they extended a generous solution by offering free replacements for any keys that were impacted by this issue, with this offer available until March 31, 2019. Furthermore, it was recommended to users to work around the problem by creating new keys externally, outside of the YubiKey, and then importing these keys onto the device. This multi-faceted approach aimed to ensure the security and satisfaction of YubiKey users.

Ultimately, the vulnerability may have affected some smart cards used for electronic payments (EMV). Chip-based payment cards utilized globally must adhere to a set of protocols outlined in the EMV standard, presently maintained by the EMV Consortium (EMVCo). The RSALib, which we have discussed, received approval for use in EMV cards from EMVCo, as corroborated by several references in related certification reports [34]. Nonetheless, it was difficult to access to publicly available datasets containing keys originating from EMV cards. In fact, from the analyses conducted by the experts, no key was found vulnerable to the attack. However, the sample at our disposal was very limited (13 cards [34]) and the results may thus not provide an entirely accurate representation of reality.

Domain	Key size (bits)	Exposure and risks
eIDs	2048	Cloning and forgery of documents
TPMs	2048	Unveil data and fake certification
PGPs	1024/2048	Forgery
EMV Smartcards	2048	Cloning and fraud transactions

Table 5.2: Summary of the possible risks with a successful factorization on a given domain. We suggest consulting [34] for further details and information.

Chapter 6

Conclusion

In the pursuit of a comprehensive understanding of Coppersmith's method and its broader implications, this master's thesis has delved into the algorithm's theoretical foundations, practical applications, and real-world consequences enlightening the algorithm's versatility and efficacy.

The relevance of the ROCA vulnerability is of utmost importance and should not be underestimated. This thesis highlights the risks associated with the utilization of shortcuts in cryptographic systems and underscores the need to adopt secure implementation methodologies. Additionally, it underscores the need for careful consideration while picking random number generators and making thorough decisions about cryptographic libraries. Furthermore, this assertion serves as a warning that no cryptographic scheme is impervious to flaws. Therefore, it's crucial to make investments in research and optimizations in order to enhance and update security as time goes on. Finally, it underscores the necessity of implementing robust patch management protocols to ensure preparedness and prompt resolution in the event of a crisis such to the one experienced by Estonia.

Appendix A

List of codes

We provide here a list of the codes used in this thesis. All codes were written by using Wolfram Mathematica 13.2. The only exception is Code A.7, which was done in SageMath and taken from `https://bitsdeep.com/posts/analysis-of-the-roca-vulnerability/`. I do not have any right on this code, all rights belong to the respective owners.

Code A.1: Howgrave-Graham Algorithm on Example 3.3.

```
(*Set given parameters*)
M = 10001;
X = 10;
a0 = -222;
a1 = 5000;
a2 = 10;
(*Build corresponding lattice*)
B = {{M, 0, 0, 0}, {0, M*X, 0, 0}, {0, 0, M*X^2, 0}, {a0, a1*X,
    a2*X^2, X^3 }};
T = LatticeReduce[B]
b0 = T[[1, 1]];
b1 = T[[1, 2]]/X;
b2 = T[[1, 3]]/X^2;
b3 = T[[1, 4]]/X^3;

F[x_] := b0 + b1*x + b2*x^2 + b3*x^3;

sol1 = Solve[F[x] == 0, x, Modulus -> M]
And@(Mod[F[x], M] == 0 /. sol1)
```

Code A.2: Extended Coppersmith on Example 3.7.

```
(*Set given parameters*)
M = (2^(20) + 7)*(2^(21) + 17);
X = 2^(9);
a0 = 227;
a1 = 46976195;
a2 = (2^(25) - 2883584);
B = {{M^2, 0, 0, 0, 0, 0, 0}, {0, M^2*X, 0, 0, 0, 0, 0}, {0, 0,
   M^2*X^2, 0, 0, 0, 0}, {M*a0, M*a1*X, M*a2*X^2, M*X^3 , 0, 0,
   0}, {0, M*a0*X, M*a1*X^2, M*a2*X^3, M*X^4, 0, 0}, {0, 0, M*a0*X^2,
   M*a1*X^3, M*a2*X^4, M*X^5, 0}, {a0^2,
   2*a0*a1*X, (a1^2 + 2*a0*a2) X^2, (2*a0 + 2*a1*a2) X^3, (a2^2 +
      2*a1) X^4, 2*a2*X^5, X^6}}
T= LatticeReduce[B]
b0 = T[[1, 1]];
b1 = T[[1, 2]]/X;
b2 = T[[1, 3]]/X^2;
b3 = T[[1, 4]]/X^3;
b4 = T[[1, 5]]/X^4;
b5 = T[[1, 6]]/X^5;
b6 = T[[1, 7]]/X^6;

F[x_] := b0 + b1*x + b2*x^2 + b3*x^3 + b4*x^4 + b5*x^5 + b6*x^6;

sol1 = Solve[F[x] == 0, x, Modulus -> M]
And @(Mod[F[x], M] == 0 /. sol1)
```

Code A.3: Bivariate Coppersmith on Exemple 3.11.

```
(*Set given parameters*)
X = 30;
Y = 20;
M = 131^4;
a = 131;
b = -1400;
c = 20;
d = -1200;
B = {{a*(X^2)*(Y^2), b*X^2*Y, c*X*Y^2, d*X*Y, 0, 0, 0, 0, 0}, {0,
    a*X^2*Y, 0, c*X*Y, b*X^2, 0, d*X, 0, 0}, {0, 0, a*X*Y^2, b*X*Y, 0,
     c*Y^2, 0, d*Y, 0}, {0, 0, 0, a*X*Y, 0, 0, b*X, c*Y,
    d}, {M*(X^2)*(Y^2), 0, 0, 0, 0, 0, 0, 0, 0}, {0, M*X^2*Y, 0, 0, 0,
     0, 0, 0, 0}, {0, 0, M*X*Y^2, 0, 0, 0, 0, 0, 0}, {0, 0, 0, M*X*Y,
    0, 0, 0, 0, 0}, {0, 0, 0, 0, M*X^2, 0, 0, 0, 0}, {0, 0, 0, 0, 0,
    M*Y^2, 0, 0, 0}, {0, 0, 0, 0, 0, 0, M*X, 0, 0}, {0, 0, 0, 0, 0, 0,
     0, M*Y, 0}, {0, 0, 0, 0, 0, 0, 0, 0, M}};
B // MatrixForm
{u, r} = HermiteDecomposition[B];
r // MatrixForm
T = r[[5 ;; 9, 5 ;; 9]]
H = LatticeReduce[T]

b0 = H[[1, 1]]/X^2;
b1 = H[[1, 2]]/Y^2;
b2 = H[[1, 3]]/X;
b3 = H[[1, 4]]/Y;
b4 = H[[1, 5]];

G[x_, y_] := b0*x^2 + b1*y^2 + b2*x + b3*y + b4
F[x_, y_] := a*x*y + b*x + c*y + d

Resultant[F[x, y], G[x, y], {x, y}] (*Find common roots*)
```

Code A.4: Partial knowledge on Example 4.4.

```
(*Set given parameters*)
M = 22461580086470571723189523;
X = 50000;
p0 = 2736273600000;
B = {{M, 0, 0, 0}, {p0, X, 0, 0}, {0, p0*X, X^2, 0}, {0, 0, p0*X^2,
    X^3}};
T=LatticeReduce[B]

b0 = T[[1, 1]];
b1 = T[[1, 2]]/X;
b2 = T[[1, 3]]/X^2;

F[x_] := b0 + b1*x + b2*x^2;

sol1 = Solve[F[x] == 0, x]

p = p0 + 27389
q = M/p
```

Code A.5: Partial knowledge with $p^r q$ on Example 4.5.

```
M = 876701170324027;
X = 10;
p0 = 5000;
F[x_] := (x + p0)^3
v = CoefficientList[F[x], x];

H = {{876701170324027, 0, 0, 0, 0, 0}, {125000000000, 75000000*X,
   15000*X^2, X^3, 0, 0}, {0, 125000000000 *X, 75000000 *X^2,
   15000 *X^3, X^4, 0}, {0, 0, 125000000000 *X^2, 75000000 *X^3,
   15000 *X^4, X^5}}

T = LatticeReduce[H]

b0 = T[[1, 1]];
b1 = T[[1, 2]]/X;
b2 = T[[1, 3]]/X^2;
b3 = T[[1, 4]]/X^3;
b4 = T[[1, 5]]/X^4;
b5 = T[[1, 6]]/X^5;

F[x_] := b0 + b1*x + b2*x^2 + b3*x^3 + b4*x^4 + b5*x^5

sol1 = Solve[F[x] == 0, x, Integers]

p = p0 + 3
q = M/p^3
```

Code A.6: Smooth Integers on Example 4.12.

```
P = 232792560;
W = 150000001;
X = 46558000;
R = -W;
B = {{P^4, 0, 0, 0, 0, 0, 0}, {-R*P^3, P^3 X, 0, 0, 0, 0,
    0 }, {R^2 P^2, -2 R*P^2 X, P^2 X^2, 0, 0, 0, 0 }, {-R^3 P,
    3 R^2 P*X, -3 R*P*X^2, P*X^3, 0, 0, 0 }, {R^4, -4 R^3 X,
    6 R^2 X^2, -4 R*X^3, X^4, 0, 0 }, {0, R^4 X, -4 R^3 X^2,
    6 R^2 X^3, -4 R*X^4, X^5, 0 }, {0, 0, R^4 X^2, -4 R^3 X^3,
    6 R^2 X^4, -4 R*X^5, X^6 }};
T = LatticeReduce[B]

b0 = T[[1, 1]];
b1 = T[[1, 2]]/X;
b2 = T[[1, 3]]/X^2;
b3 = T[[1, 4]]/X^3;
b4 = T[[1, 5]]/X^4;

F[x_] := b0 + b1*x + b2*x^2 + b3*x^3 + b4*x^4

sol1 = Solve[F[x] == 0, x]
And@(Mod[F[x], M] == 0 /. sol1)
FactorInteger[W - 33603721]
```

Code A.7: Flawed prime generation and factorization[1]

```
def primorial(n):
    M = 1
    p = 1
    for i in range(n):
        p = next_prime(p)
        M *= p
    return int(M)

def getParameterSizes(keySize):
    if 512 <= keySize <= 960:
        n = 39
    elif 992 <= keySize <= 1952:
        n = 71
    elif 1984 <= keySize <= 3936:
        n = 126
    elif 3968 <= keySize <= 4096:
        n = 225
    else:
        print("Invalid key size.")
        return None
    M = primorial(n)
    k_size = keySize//2 - round(log(M, 2))
    a_size = ceil(log(Zmod(M)(65537).multiplicative_order(), 2))
    return M, k_size, a_size

def genWeakPrime(M, k_size, a_size):
    from Crypto.Util.number import getRandomNBitInteger
    p = 0
    while not is_prime(p):
        k = getRandomNBitInteger(k_size)
        a = getRandomNBitInteger(a_size)
        p = k*M + pow(int(65537), a, M)
    return p
```

[1]Taken from `https://bitsdeep.com/posts/analysis-of-the-roca-vulnerability/` who adapted the code from `https://github.com/mimoo/RSA-and-LLL-attacks/blob/master/coppersmith.sage`. All rights belong to the respective owners.

```
def genWeakKey(size):
    params = getParameterSizes(size)
    if params is not None:
        M, k_size, a_size = params
        n = int(0)
        # Can happen sometimes
        while n.bit_length() != size:
            p1 = genWeakPrime(M, k_size, a_size)
            p2 = genWeakPrime(M, k_size, a_size)
            n = p1*p2
        return n

n = genWeakKey(512)
print(n)
print(n.bit_length())

def coppersmith_howgrave_univariate(pol, modulus, beta, mm, tt, XX):

    dd = pol.degree()
    nn = dd * mm + tt

    if not 0 < beta <= 1:
        raise ValueError("beta should belongs in (0, 1]")

    if not pol.is_monic():
        raise ArithmeticError("Polynomial must be monic.")

    polZ = pol.change_ring(ZZ)
    x = polZ.parent().gen()

    gg = []
    for ii in range(mm):
        for jj in range(dd):
            gg.append((x * XX)**jj * modulus**(mm - ii) * polZ(x * XX)**ii)
    for ii in range(tt):
        gg.append((x * XX)**ii * polZ(x * XX)**mm)

    # construct lattice B
    BB = Matrix(ZZ, nn)
```

```
    for ii in range(nn):
        for jj in range(ii+1):
            BB[ii, jj] = gg[ii][jj]

    # LLL
    BB = BB.LLL()

    # transform shortest vector in polynomial
    new_pol = 0
    for ii in range(nn):
        new_pol += x**ii * BB[0, ii] / XX**ii

    # factor polynomial
    potential_roots = new_pol.roots()

    # test roots
    roots = []
    for root in potential_roots:
        if root[0].is_integer():
            result = polZ(ZZ(root[0]))
            if gcd(modulus, result) >= modulus^beta:
                roots.append(ZZ(root[0]))

    return roots

    def solve(M, n, a, m, t, beta):
      base = int(65537)
    # the known part of p: 65537^a * M^-1 (mod N)
    known = int(pow(base, a, M) * inverse_mod(M, n))
    # Create the polynomial f(x)
    F = PolynomialRing(Zmod(n), implementation='NTL', names=('x',))
    (x,) = F._first_ngens(1)
    pol = x + known
    # Upper bound for the small root x0
    XX = floor(2 * n^0.5 / M)
    # Find a small root (x0 = k) using Coppersmith's algorithm
    roots = coppersmith_howgrave_univariate(pol, n, beta, m, t, XX)
    # There will be no roots for an incorrect guess of a.
    for k in roots:
        # reconstruct p from the recovered k
```

```
        p = int(k*M + pow(base, a, M))
        if n%p == 0:
            return p, n//p

M, k_s, a_s = getParameterSizes(512)
p = genWeakPrime(M, k_s, a_s)
q = genWeakPrime(M, k_s, a_s)
n = p*q
print(f"n: {n}")
print(f"p: {p}")
print(f"q: {q}")
# take the biggest factor because Beta = 0.5, it will not work on the
#other
f = max(p, q)

# Compute the right values for a and k
a = int(Zmod(M)(f).log(65537))
print(f"a: {a}")
k = (f - int(pow(65537, a, M))) // M
print(f"k: {k}")
# Assert correctness of the values
assert(k*M + pow(65537, a, M) == f)
# solve using the parameters from earlier
factors = solve(M, n, a, 5, 6, 0.5)
print(f"factorized key with right guess: {factors}")
```

Acronyms

CRT	Chinese Remainder Theorem.
CRTLDP	CRT List Decoding Problem.
CVE	Common Vulnerabilities and Exposures.
DLP	Discrete Logarithm Problem.
ECC	Elliptic Curves Cryptography.
ECM	Elliptic Curves Methods.
eID	Electronic Identification.
EK	Endorsement Key.
EMV	Electronic Payment Card/Europay, Mastercard and Visa.
EMVCo	EMV Consortium.
FPS	Fixed Padding Scheme.
HG	Howgrave-Graham.
HSM	Hardware Security Modules.
LLL	Lenstra–Lenstra–Lovász.
PGP	Pretty Good Privacy.
PPA	Police and Border Guard.
ROCA	Return of Coppersmith's Attack.
RP	Random Padding.
RSA	Rivest–Shamir–Adleman.
SRK	Storage Root Key.
SVP	Shortest Vector Problem.
TPM	Trusted Platform Modules.

Bibliography

[1] Statistics about Internet voting in Estonia — Elections in Estonia. URL https://www.valimised.ee/en/archive/statistics-about-internet-voting-estonia.

[2] The Trusted Platform Module Key Hierarchy. URL https://ericchiang.github.io/post/tpm-keys/.

[3] ROCA Vulnerability and eID: Lessons Learned, May 2018. URL https://www.ria.ee/en/media/742/download.

[4] Miklós Ajtai. The Shortest Vector Problem in L2 is NP-Hard for Randomized Reductions (Extended Abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 10–19, New York, NY, USA, 1998. Association for Computing Machinery. URL https://doi.org/10.1145/276698.276705.

[5] Aurélie Bauer and Antoine Joux. Toward a rigorous variation of coppersmith's algorithm on three variables. In *Proceedings of the 26th Annual International Conference on Advances in Cryptology*, EUROCRYPT '07, page 361–378, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 9783540725398. doi: 10.1007/978-3-540-72540-4_21. URL https://doi.org/10.1007/978-3-540-72540-4_21.

[6] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, pages 92–111, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. ISBN 978-3-540-44717-7.

[7] Daniel J. Bernstein and Tanja Lange. The cr.yp.to blog, May 2017. URL https://blog.cr.yp.to/20171105-infineon.html.

[8] Dan Boneh. Twenty Years Of Attacks On The RSA Cryptosystem. *Notices of the American Mathematical Society*, 46:203–212, 1999. URL https://api.semanticscholar.org/CorpusID:11023027.

[9] Dan Boneh. Finding Smooth Integers in Short Intervals Using CRT Decoding. *Journal of Computer and System Sciences*, 64(4):768–784, 2002. ISSN 0022-0000. doi: https://doi.org/10.1006/jcss.2002.1827. URL https://www.sciencedirect.com/science/article/pii/S0022000002918271.

[10] Dan Boneh, Glenn Durfee, and Nick Howgrave-Graham. *Factoring $N = p^r q$ for Large r*. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 326–337, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-48405-9.

[11] Ted Chinburg, Brett Hemenway, Nadia Heninger, and Zachary Scherr. Cryptographic applications of capacity theory: On the optimality of Coppersmith's method for univariate polynomials, 2016.

[12] Don Coppersmith. Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. pages 178–189, 05 1996. ISBN 978-3-540-61186-8. doi: 10.1007/3-540-68339-9_16.

[13] Don Coppersmith. Finding a Small Root of a Univariate Modular Equation. volume 1070, pages 155–165, 05 1996. ISBN 978-3-540-61186-8. doi: 10.1007/3-540-68339-9_14.

[14] Don Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *Journal of Cryptology*, 10:233–260, 1997.

[15] Don Coppersmith. *Finding Small Solutions to Small Degree Polynomials*, pages 20–31. 09 2001. ISBN 978-3-540-42488-8. doi: 10.1007/3-540-44670-2_3.

[16] Don Coppersmith, Matthew Franklin, Jacques Patarin, and Michael Reiter. Low-Exponent RSA with Related Messages. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 1–9, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. ISBN 978-3-540-68339-1.

[17] Jean-Sébastien Coron. Finding Small Roots of Bivariate Integer Polynomial Equations: A Direct Approach. CRYPTO'07, page 379–394, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3540741429.

[18] Jean-Sébastien Coron, Jean-Charles Faugère, Guénaël Renault, and Rina Zeitoun. Factoring $N = p^r q^s$ for Large r and s. In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016*, pages 448–464, Cham, 2016. Springer International Publishing. ISBN 978-3-319-29485-.

[19] Jean-Sébastien Coron, Alexey Kirichenko, and Mehdi Tibouchi. A Note on the Bivariate Coppersmith Theorem. *Journal of Cryptology*, 26:246–250, 2013. doi: 10.1007/s00145-012-9121-x.

[20] Donald Davies. A Brief History of Cryptography. *Information Security Technical Report*, 2(2):14–17, 1997. ISSN 1363-4127. doi: https://doi.org/10.1016/S1363-4127(97)81323-4.

[21] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[22] Enoent. Analysis of the ROCA vulnerability, Apr 2020. URL `https://bitsdeep.com/posts/analysis-of-the-roca-vulnerability/`.

[23] Matthew K Franklin. A linear protocol failure for RSA with exponent three. *Crypto'95 Rump Session, August*, 1995.

[24] Atsushi Fujioka, Tatsuaki Okamoto, and Shoji Miyaguchi. ESIGN: An Efficient Digital Signature Implementation for Smart Cards. In Donald W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, pages 446–457, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. ISBN 978-3-540-46416-7.

[25] Steven D. Galbraith. *Mathematics of Public Key Cryptography.* Cambridge University Press, USA, 1st edition, 2012. ISBN 1107013925.

[26] Steven D. Galbraith. *Coppersmith's method and related applications*, page 380–402. Cambridge University Press, 2012. doi: 10.1017/CBO9781139012843.020.

[27] Oded Goldreich, Dana Ron, and Madhu Sudan. Chinese Remaindering with Errors. Cryptology ePrint Archive, Paper 1999/002, 1999. URL `https://eprint.iacr.org/1999/002`. `https://eprint.iacr.org/1999/002`.

[28] Kaspar Korjus. Digital ID cards now only work with new certificates, Nov 2017. URL `https://medium.com/e-residency-blog/digital-id-cards-will-only-work-with-new-certificates-from-midnight-tonight-6e4db59658b8`.

[29] Arjen K. Lenstra, Jr. Hendrik W. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. ISSN 0025–5831. URL: `http://cr.yp.to/bib/entries.html#1982/lenstra-lll`.

[30] Alexander May. *New RSA vulnerabilities using lattice reduction methods.* PhD thesis, Citeseer, 2003.

[31] Alexander May. Using LLL-reduction for solving RSA and factorization problems. In *The LLL Algorithm: Survey and Applications*, pages 315–348. Springer, 2009.

[32] Alexander May. *Lattice-Based Integer Factorisation: An Introduction to Coppersmith's Method*, page 78–105. London Mathematical Society Lecture Note Series. Cambridge University Press, 2021. doi: 10.1017/9781108854207.006.

[33] Majid Mumtaz and Ping Luo. Cryptanalysis of RSA: A Special Case of Boneh-Durfee's Attack. Cryptology ePrint Archive, Paper 2020/1214, 2020. URL `https://eprint.iacr.org/2020/1214`.

[34] Matus Nemec, Marek Sys, Petr Svenda, Dusan Klinec, and Vashek Matyas. The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli. In *24th ACM Conference on Computer and Communications Security (CCS'2017)*, pages 1631–1648. ACM, 2017. ISBN 978-1-4503-4946-8/17/10.

[35] Phong Q. Nguyen and Jacques Stern. The Two Faces of Lattices in Cryptology. In Joseph H. Silverman, editor, *Cryptography and Lattices*, pages 146–180, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[36] Sean Ong. Evaluating the Security of National Electronic Identification Programs. COMP 116: Introduction to Computer Security, 2018. URL https://www.cs.tufts.edu/comp/116/archive/spring2018/song.pdf.

[37] Arnis Parsovs. Solving the Estonian ID Card Crisis: the Legal Issues. In *International Conference on Information Systems for Crisis Response and Management*, 2020. URL https://api.semanticscholar.org/CorpusID:224816143.

[38] Bruno Produit. Optimization of the ROCA (CVE-2017-15361) Attack. *Master's thesis, UNIVERSITY OF TARTU, Institute of Computer Science, Estonia*, 2019.

[39] Adam Rang. How does smart-ID benefit e-residents?, Jun 2020. URL https://medium.com/e-residency-blog/how-does-smart-id-benefit-e-residents-6c5d3d6bed4.

[40] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[41] Joachim Rosenthal. Kryptographie Lecture Notes, Autumn 2011.

[42] Claus-Peter Schnorr. A more efficient algorithm for lattice basis reduction. *Journal of Algorithms*, 9(1):47–62, 1988. ISSN 0196-6774. doi: https://doi.org/10.1016/0196-6774(88)90004-1.

[43] Claus-Peter Schnorr. Gittertheorie und algorithmische Geometrie, Reduktion von Gitterbasen und Polynomidealen, Summer 1994.

[44] Panda Security. PGP encryption: The email security standard - panda security, Jun 2023. URL https://www.pandasecurity.com/en/mediacenter/panda-security/pgp-encryption/.

[45] Isabel Skierka. When shutdown is no option: Identifying the notion of the digital government continuity paradox in Estonia's eID crisis. *Government Information Quarterly*, 40(1):101781, 2023. ISSN 0740-624X. doi: https://doi.org/10.1016/j.giq.2022.101781. URL https://www.sciencedirect.com/science/article/pii/S0740624X22001174.

[46] Astrid Valtna-Dvořák, Silvia Lips, Valentyna Tsap, Rain Ottis, Jaan Priisalu, and Dirk Draheim. Vulnerability of State-Provided Electronic Identification: The Case of ROCA in Estonia. In Andrea Kö, Enrico Francesconi, Gabriele Kotsis, A. Min Tjoa, and Ismail Khalil, editors, *Electronic Government and the Information Systems Perspective*, pages 73–85, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86611-2.

[47] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra:* . Modern Computer Algebra. Cambridge University Press, 2013. ISBN 9781107039032. URL https://books.google.ch/books?id=7fE9baKyqSEC.

[48] Robin Wilton. ROCA: Encryption vulnerability and what to do about it, Oct 2017. URL https://www.internetsociety.org/blog/2017/11/roca-encryption-vulnerability/.