Diss. ETH No. 11404

# Efficiency and Security of Cryptosystems based on Number Theory

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZÜRICH

for the degree of
Doctor of Technical Sciences

presented by

DANIEL BLEICHENBACHER
Dipl. Inform. Universität Bern

born August 4, 1964
citizen of Gottshaus, TG

accepted on the recommendation of

Prof. Dr. U. Maurer, referee
Prof. Dr. M. Bronstein, co-referee
Dr. A. K. Lenstra, co-referee

Zürich, 1996

# Acknowledgments

I am deeply grateful to Prof. Ueli Maurer for guiding this work and showing me the pleasure of doing research. Prof. Manuel Bronstein has made numerous helpful comments and suggestions, which are gratefully acknowledged. A large note of appreciation goes to Dr. Arjen Lenstra for accepting to be a co-referee for this thesis. It was a great pleasure working on a joint paper with him. I would also like to thank Prof. Roman Mäder for being a referee for the qualifying examination and Dr. Wieb Bosma for some corrections. Many thanks go to my friends at the information security and cryptography research group at ETH, Markus Stadler, Christian Cachin, Jan Camenisch, Stefan Wolf and Martin Hirt, for providing an inspirational environment.

# Abstract

Computational number theory plays an important role in cryptography because many cryptographic systems and protocols are based on algebraic and number-theoretic structures. Among the important number-theoretic problems relevant to cryptography are primality testing, factoring integers, and discrete logarithms in finite groups.

Efficiency and security are two natural but conflicting goals in cryptography. This dissertation is concerned with a number of security and efficiency aspects of cryptosystems based on number theory. It consists of three parts: contributions to primality testing, to the problem of finding optimal addition chains and Lucas chains, and to the security analysis of a number of proposed cryptographic schemes. Several pseudo-primality tests are analyzed and compared and algorithms for efficiently finding all counter-examples up to a given bound are described. A fundamental cryptographic operation is the exponentiation in finite groups. An exponentiation corresponds to an addition chain for the exponent. An efficient algorithm for computing shortest addition chains is described. The closely related problem of computing Lucas chains is also considered and an algorithm for computing shortest Lucas chains is described. Finally, some weaknesses in proposed cryptosystems are presented. It is shown that ElGamal signatures can be forged in certain cases and a new chosen message attack against a Lucas-based analogue of the RSA system is described.

# Zusammenfassung

Algorithmen der Zahlentheorie spielen eine wichtige Rolle in der Kryptographie, da viele kryptographische Systeme und Protokolle auf algebraischen und zahlentheoretischen Strukturen basieren. Zu den wichtigen zahlentheoretischen Problemen, die für die Kryptographie relevant sind, gehören Primzahltests, Faktorisierung von ganzen Zahlen und die Berechnung von diskreten Logarithmen in endlichen Gruppen.

Effizienz und Sicherheit sind zwei natürliche aber widersprüchliche Ziele in der Kryptographie. Diese Dissertation befasst sich mit einigen Aspekten zur Sicherheit und Effizienz von Kryptosystemen, die auf der Zahlentheorie basieren. Sie besteht aus drei Teilen: Primzahltests, Algorithmen um kürzeste Additions- und Lucas-Ketten zu finden und Analysen einiger vorgeschlagener Kryptosysteme. Verschiedene Pseudo-Primzahltests werden analysiert und verglichen. Algorithmen, die alle Gegenbeispiele zu Pseudo-Primzahltests bis zu einer vorgegebenen Grenze finden, werden beschrieben. Exponentiation ist eine grundlegende Operation in der Kryptographie. Eine Exponentiation entspricht einer Additionskette für dessen Exponenten. Ein Algorithmus, der kürzeste Additionsketten berechnet, wird beschrieben. Betrachtet wird ebenfalls das eng verwandte Problem, kürzeste Lucas Ketten zu berechnen. Zum Schluss werden einige Schwachpunkte in vorgeschlagenen Kryptosystemen vorgestellt. Es wird gezeigt, dass ElGamal Unterschriften in gewissen Fällen gefälscht werden können. Eine neue "chosen-message"-Attacke gegen ein auf den Lucas Funktionen basierendes Analogon des RSA Systemes wird vorgestellt.

# Contents

# Chapter 1

# Introduction

Cryptography is one of the most applied disciplines within theoretical computer science. Among its applications are data encryption, key management, digital signatures, user authentication, digital payment systems and a large number of protocols for performing tasks like, for instance, electronic voting. Cryptography is one of the basic technologies for building the emerging international information infrastructure, in particular for supporting electronic commerce and other security-relevant services.

A particularly important field within cryptography is public-key cryptography. Secret keys in public-key cryptographic mechanisms need to be stored only at a single location. This is in contrast to conventional or symmetric cryptography where the secret keys are stored at least at two different locations (for instance for encryption and decryption). This is the major reason why public-key cryptography is of great importance in key management and other applications.

Only a small number of mathematical structures could so far be applied to build public-key mechanisms, and the majority of these structures is based on number theory and algebra, in particular on the multiplicative group of integers modulo a large number. As a consequence, computational number theory plays an important role in cryptography.

The security of no presently-used cryptographic system can be proved. Such a proof would be a major breakthrough in cryptography and in complexity theory because it would imply a significant lower bound on the complexity of a computational problem. It is well-known that no computational problem, whether of cryptographic nature or not, has so far been proved to be difficult in a general computational model. The closest to a security proof one can achieve is that a certain cryptosystem is proved to be as difficult to break as the solution of a well-known computational problem that is conjectured to be difficult (e.g. factoring large integers or computing discrete logarithms in a finite group).

This dissertation is concerned with a number of security and efficiency aspects of cryptosystems based on number theory. It consists of three parts: contributions to primality testing, to the problem of finding optimal addition chains and Lucas chains, and to the security analysis of a number of proposed cryptographic schemes.

Most modern cryptosystems make use of large prime numbers, either as secret or as public parameters. The generation of large prime numbers is hence a basic cryptographic operation. One way of generating a large prime is by choosing a large number at random and testing it for primality. There exist several sophisticated general-purpose algorithms for testing primality but their efficiency is not sufficient for most applications. In practise one therefore resorts to *pseudo-primality* tests, thus tolerating a small probability of letting a composite number pass as a prime. Primality testing is also of independent interest in computer algebra. One of the contributions of this dissertation, described in Chapter 3, is a systematic analysis of previously proposed pseudo-primality tests. The principal part of this chapter describes two algorithms for finding counterexamples to pseudo-primality tests.

Exponentiation in a finite group is a fundamental cryptographic operation. The problem of minimizing the number of multiplications for raising an element to a given power $n$ is equivalent to finding an optimal addition chain for $n$, that is one of minimal length. An addition chain for $n$ is a finite ascending sequence of integers starting with 1 and ending with $n$ such that every element of the sequence except the first is the sum of two preceding elements. It is remarkable that Knuth devoted about 20 pages of his book "The Art of Computer Programming; Seminumerical Algorithms" to this problem. In Chapter 4, several results on addition chains are proved and these results are combined to obtain an efficient algorithm for finding the shortest addition chain for all numbers below 500000.

Chapter 5 addresses a problem related to addition chains, namely finding optimal Lucas chains. While short addition chains lead to efficient exponentiation algorithms, short Lucas chains lead to the efficient computation of elements of Lucas sequences, which are used in a number of proposed cryptosystems.

Chapter 6 is concerned with the analysis of cryptosystems. In the first part of the chapter a new method for forging ElGamal signatures is demonstrated. This attack is possible if the public parameters of the system are not chosen with care. Because it does not recover the secret key this attack demonstrates that forging ElGamal signatures is sometimes easier than the underlying discrete logarithm problem. Moreover, a proposed variant of ElGamal's signature scheme is analyzed. It is shown that this version is much weaker than believed. The second part of the chapter is devoted to a Lucas-based analogue of the RSA scheme. It has been claimed that this scheme is stronger than RSA, because no chosen message attack for this scheme is possible. A novel attack which needs only one chosen message will be shown.

# Chapter 2

# Mathematical preliminaries

## 2.1 Symbols and Notation

$\mathbb{Z}/(m)$ denotes the ring of integers modulo $m$. If $R$ is a ring then $R^*$ denotes the *group of units* of $R$. (i.e. $R^*$ is the multiplicative group of all elements $x \in R$ such that $xy = 1$ for some $y \in R$.) If $p^\alpha$ is a prime power then $\mathbb{F}_{p^\alpha}$ denotes the field of order $p^\alpha$.

## 2.2 Quadratic extensions

For $\Delta$ not a perfect square $\mathbb{Q}(\sqrt{\Delta})$ denotes the subfield of $\mathbb{C}$ consisting of the numbers $\{r + s\sqrt{\Delta} : r, s \in \mathbb{Q}\}$. $\mathbb{Q}(\sqrt{\Delta})$ is a *real extension* of $\mathbb{Q}$ if $\Delta > 0$ and $\mathbb{Q}(\sqrt{\Delta})$ is a *complex extension* of $\mathbb{Q}$ if $\Delta < 0$.

In $\mathbb{Q}(\sqrt{\Delta})$ we define the *conjugate* of $r + s\sqrt{\Delta}$ by

$$\overline{r + s\sqrt{\Delta}} := r - s\sqrt{\Delta}.$$

If $\mathbb{Q}(\sqrt{\Delta})$ is a complex extension then the conjugate is equivalent to the conjugate in the complex numbers for any embedding of $\mathbb{Q}(\sqrt{\Delta})$ in $\mathbb{C}$. The two definition of conjugation are, however, not equivalent when $\mathbb{Q}(\sqrt{\Delta})$ is a real extension. The *trace* $\mathrm{tr}(x)$ is defined by

$$\mathrm{tr}(x) = x + \overline{x}$$

i.e. $\mathrm{tr}(r + s\sqrt{\Delta}) = 2r$. The *norm* $\mathrm{n}(x)$ is defined by

$$\mathrm{n}(x) = x \cdot \overline{x}$$

$\mathcal{O}_{\sqrt{\Delta}}$ is the ring of integers of the quadratic extension $\mathbb{Q}(\sqrt{\Delta})$, i.e. the set of all numbers in $\mathbb{Q}(\sqrt{\Delta})$ that are roots of an equation $x^2 + bx + c$ with $b, c \in \mathbb{Z}$.

If $\Delta$ is squarefree then

$$\mathcal{O}_{\sqrt{\Delta}} = \begin{cases} \{r + s\sqrt{\Delta} : r, s \in \mathbb{Z}\} & \text{if} \quad \Delta \equiv 2 \text{ or } 3 \pmod 4 \\ \{r + s\frac{-1+\sqrt{\Delta}}{2} : r, s \in \mathbb{Z}\} & \text{if} \quad \Delta \equiv 1 \pmod 4). \end{cases}$$

For $a, b \in \mathcal{O}_{\sqrt{\Delta}}$ we have $a \equiv b \pmod n$ if and only if $a - b \in n\mathcal{O}_{\sqrt{\Delta}}$. We refer to [IR90, chapter 13] for more details.

## 2.3 Number-theoretic functions

**Definition 2.1** *For all $n \geq 1$ with prime factorization $n = \prod_{i=1}^{r} p_i^{\alpha_i}$ we define the following functions:*

$\tau(n)$ *is the number of divisors of $n$, i.e. $\tau(n) = \sum_{d|n} 1$.*

$\omega(n)$ *is the number of distinct prime divisors of $n$, i.e. $\omega(n) = r$.*

$\beta(n)$ *is the number of prime factors with multiplicity of $n$, i.e. $\beta(n) = \sum_{i=1}^{r} \alpha_i$.*

$\nu_p(n)$ *is the maximal power of $p$ in $n$, i.e. $\nu_p(n) = \max\{i : p^i$ divides $n\} = \alpha_i$.*

$\sigma(n)$ *is the sum of the divisors of $n$, i.e. $\sigma(n) = \sum_{d|n} d$.*

$\varphi(n)$ *is the number of positive integer $< n$ relatively prime to $n$, i.e. $\varphi(n) = \#\{x : x < n$ and $\gcd(x, n) = 1\}$*

$\lambda(n)$ *is the* Carmichael function. $\lambda(p^\alpha) := (p-1)p^{a-1}$ *if $p^\alpha$ is a prime power and $p \neq 2$. $\lambda(2) = 1, \lambda(4) = 2$ and $\lambda(2^\alpha) = 2^{\alpha-2}$. For composite numbers* $\lambda(n) := \mathrm{lcm}(\lambda(p_1^{\alpha_1}), \ldots, \lambda(p_r^{\alpha_r}))$.

$\pi(x)$ *The number of primes $p < x$.*

**Theorem 2.2** *For $n = \prod_{i=1}^{r} p_i^{\alpha_i}$ we have*

$$\tau(n) = \prod_i (\alpha_i + 1) \tag{2.1}$$

$$\sigma(n) = \prod_i \frac{p_i^{\alpha_i+1} - 1}{p_i - 1} = n \prod_i \left(1 + \frac{1 - p_i{}^{-\alpha_i}}{p_i - 1}\right) \tag{2.2}$$

$$\varphi(n) = \prod_i p_i^{\alpha_i - 1}(p_i - 1) = n \prod_i (1 - p_i^{-1}) \tag{2.3}$$

Proofs for these equations can be found in almost any book on number theory. (see for example [IR90, chapter 2])

**Theorem 2.3 (Prime number theorem)**

$$\pi(x) \approx x / \ln(x) \tag{2.4}$$

The prime number theorem has first been proved by Hadamard and C. de la Vallée Poussin independently. A simpler proof of the theorem was found by Newman [New80]. An extensive overview is given in [Dia82].

**Theorem 2.4**
$$\frac{n}{\varphi(n)} = O(\ln\ln(n)) \tag{2.5}$$

This theorem can for example be used to estimate the number of generators in a cyclic group. We refer to [HW79] for a proof.

## 2.4 Chinese Remainder Theorem (CRT)

**Theorem 2.5** *Let $m = m_1 m_2 \cdots m_r$ such that the $m_i$'s are pairwise relatively prime and let $b_1, b_2, \ldots, b_r$ be integers. Then the system*

$$\begin{array}{rcl} x & \equiv & b_1 \pmod{m_1} \\ x & \equiv & b_2 \pmod{m_2} \\ & \cdots & \\ x & \equiv & b_r \pmod{m_r} \end{array}$$

*is solvable and has a unique solution modulo $m$.*

*Proof:* Let $n_i = m/m_i$ and $s_i \equiv n_i^{-1} \pmod{m_i}$. Note that the inverse of $n_i$ exists, since $\gcd(n_i, m_i) = 1$. Let $x' = \sum_{i=1}^r b_i s_i n_i$. We have $n_j \equiv 0 \pmod{m_i}$ for all $i \neq j$ and therefore $x' \equiv \sum_{i=1}^r b_i s_i n_i \equiv b_j s_j n_j \equiv b_j \pmod{m_j}$. Thus $x'$ is a solution of the system.

Now let $x$ and $x'$ be two solutions. Then we have $x - x' \equiv 0 \pmod{m_i}$ for all $i$. Thus $m_i$ divides $x - x'$ for $1 \leq i \leq r$. From $\mathrm{lcm}(m_1, m_2 \ldots, m_r) = m$ follows therefore that $x - x' \equiv 0 \pmod{m}$. □

*Remark.* Note that the proof is constructive. It shows a method to find the solution in time $O(\ln(m)^2)$ (assuming schoolbook multiplication and the use of Euclid's algorithm for the computation of modular inverses).

## 2.5 The Legendre-Jacobi symbol

In this section we define the Legendre and Jacobi symbols. These two symbols define essentially the same function. The only difference is the domain for which these functions are defined.

For a prime $p$ and an integer $a$ the *Legendre symbol* $\left(\frac{a}{p}\right)$ is defined by

$$\left(\frac{a}{p}\right) := \begin{cases} 0 & \text{if} \quad \gcd(a,p) > 1 \\ 1 & \text{if} \quad a \equiv x^2 \pmod{p} \text{ for some integer } x \\ -1 & \quad\text{otherwise} \end{cases}$$

The Legendre symbol can be computed by (2.6), which was discovered by Euler. An even more efficient method to compute the Legendre symbol is to use the Jacobi symbol, which is defined in the sequel.

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod p \tag{2.6}$$

The *Jacobi symbol* $\left(\frac{a}{m}\right)$ is an extension of the Legendre symbol to all odd positive $m \in \mathbb{Z}$. Let $m = p_1 \cdots p_r$ with $p_i$ prime. Then $\left(\frac{a}{m}\right)$ is defined by

$$\left(\frac{a}{m}\right) := \begin{cases} 0 \text{ if } \gcd(a,m) > 1 \\ \left(\frac{a}{p_1}\right) \cdots \left(\frac{a}{p_r}\right) \text{ otherwise} \end{cases}$$

where $\left(\frac{a}{p_i}\right)$ denotes the corresponding Legendre symbol.

An extension of the Jacobi symbol to all $m \in \mathbb{Z}$, which is called Kronecker symbol, can for example be found in [Coh93, Section 1.4.2]. However, since some of the most important properties of the Jacobi symbol (see Theorem 2.6) are no longer true for the Kronecker symbol we will not use this symbol.

**Theorem 2.6** *Let $a, b, k \in \mathbb{Z}$ and let $m, n$ be odd positive integers.*

$$\left(\frac{ab}{m}\right) = \left(\frac{a}{m}\right)\left(\frac{b}{m}\right) \tag{2.7}$$

$$\left(\frac{a}{mn}\right) = \left(\frac{a}{m}\right)\left(\frac{a}{n}\right) \tag{2.8}$$

$$\left(\frac{2}{m}\right) = (-1)^{(m^2-1)/8} \tag{2.9}$$

$$\left(\frac{a}{m}\right) = \left(\frac{a+km}{m}\right) \tag{2.10}$$

$$\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)(-1)^{(m-1)(n-1)/4} \tag{2.11}$$

$$\left(\frac{a}{m+4ak}\right) = \left(\frac{a}{m}\right) \quad \textit{if } m + 4ak > 0 \tag{2.12}$$

*Proof:* We refer to [IR90, Proposition 5.2.1 and 5.2.2] for proofs of (2.7)–(2.11). It remains to prove (2.12). Because of (2.10) we may assume that $a$ is positive. Since $\left(\frac{ab}{m}\right) = \left(\frac{a}{m}\right)\left(\frac{b}{m}\right)$ it is sufficient to show that the lemma is correct for $a = 2$ and $a$ odd. If $a = 2$ then

$$\left(\frac{2}{m+8k}\right) = (-1)^{((m+8k)^2-1)/8} = (-1)^{(m^2-1)/8} = \left(\frac{2}{m}\right).$$

For $a$ odd we have by (2.11) and (2.12)

$$\left(\frac{a}{m+4ak}\right) = \left(\frac{m+4ak}{a}\right)(-1)^{\frac{(a-1)(m+4ak-1)}{4}} =$$

$$\left(\frac{m}{a}\right)(-1)^{\frac{(a-1)(m+4ak-1)}{4}} = \left(\frac{a}{m}\right)(-1)^{\frac{(a-1)(m+4ak-1)}{4}}(-1)^{\frac{(a-1)(m-1)}{4}} =$$
$$\left(\frac{a}{m}\right)(-1)^{(a-1)ak} = \left(\frac{a}{m}\right). \quad \square$$

*Remarks.* Using Theorem 2.6 it is possible to compute $\left(\frac{m}{n}\right)$ in polynomial time (i.e. $O(\ln(n)^2)$ if $|m| < n$). The crucial equation is 2.11, since this equation allows the computation of $\left(\frac{m}{n}\right)$ without knowing the factorization of $n$. A detailed description of an algorithm can be found in [Coh93, Section 1.4.2].

So far no algorithm is known to decide, whether $a$ is a quadratic residue modulo $n$ without factoring $n$ and it is not known whether deciding if an integer is a quadratic residue modulo $n$ is equally hard as factoring $n$ [AM94, C 11].

**Theorem 2.7** *Let $p$ be an odd prime, $q = p^\alpha$ be a prime power and $b \in \mathbb{Z}/(q)^*$. Then*

$$x^2 \equiv b \pmod{q}$$

*is solvable if and only if* $\left(\frac{b}{p}\right) = 1$.

*Proof:* If $x^2 \equiv b \pmod{q}$ is solvable then $x^2 \equiv b \pmod{p}$ is solvable too. Hence $\left(\frac{b}{p}\right) = 1$ follows by definition.

Now assume that $\left(\frac{b}{p}\right) = 1$. We will prove that $x^2 \equiv b \pmod{p^\alpha}$ is solvable by induction over $\alpha$. It follows from the definition of the Legendre symbol that $x^2 \equiv b \pmod{p}$ is solvable. Now assume that $x^2 \equiv b \pmod{p^\alpha}$ for $\alpha \geq 1$. Let $k = (b - x^2)/p^\alpha$. It follows that

$$(x + kp^\alpha 2^{-1})^2 \equiv x^2 + kp^\alpha + k^2 p^{2\alpha} 4^{-1} \equiv x^2 + kp^\alpha \equiv b \pmod{p^{\alpha+1}}.$$

Hence $x^2 \equiv b \pmod{p^{\alpha+1}}$ is solvable too. $\square$

## 2.6 The order of a group element

**Definition 2.8** *Let $g$ be an element of a finite group $(G, \cdot)$. The order of $g$, denoted by $\mathrm{ord}_G(g)$, is the smallest exponent $t > 0$ satisfying $g^t = 1$ where $1$ is the neutral element of $G$.*

Alternatively, the order of an element $g$ can be defined as the cardinality of the subgroup generated by $g$. We will often compute orders of elements $x$ in $\mathbb{Z}/(m)^*$ and therefore abbreviate the notation for such orders by $\mathrm{ord}_m(x)$. The following result is due to Lagrange.

**Theorem 2.9** *The order of an element divides the cardinality of the group.*

Lagrange's Theorem implies for $\mathbb{Z}/(m)^*$ for all $b \in \mathbb{Z}/(m)^*$ that

$$\begin{aligned} & \operatorname{ord}_m(b) \quad \text{divides} \quad \varphi(m) && (2.13)\\ \text{and} \quad & \operatorname{ord}_m(b) \quad \text{divides} \quad \lambda(m). && (2.14)\end{aligned}$$

We refer to [Rie85, p.276] for a proof.

**Theorem 2.10** *Let $p \neq 2$ be a prime, $q = p^\alpha$ be a prime power, $g$ be a generator of $\mathbb{Z}/(q)^*$ and $b \equiv g^r \pmod{q}$. Then*

*(i)* $\operatorname{ord}_q(b) = \frac{\varphi(q)}{\gcd(r,\varphi(q))}$,

*(ii)* $\nu_2(\operatorname{ord}_q(b)) = \nu_2(p-1)$ *if and only if* $\left(\frac{b}{p}\right) = -1$,

*(iii)* $\operatorname{ord}_q(b) = \operatorname{ord}_p(b)p^t$ *for some* $0 \leq t \leq \alpha - 1$.

*Proof:* (i) By definition $\operatorname{ord}_q(b)$ is the smallest exponent $i > 0$ satisfying $b^i \equiv g^{ir} \equiv 1 \pmod{q}$. Since $g$ is a generator of $\mathbb{Z}/(q)^*$ this is equivalent to

$$ir \equiv 0 \pmod{\varphi(q)}.$$

This equation is satisfied if and only if $i$ is a multiple of $\varphi(q)/\gcd(r,\varphi(q))$.

(ii) It follows from (i) that

$$\begin{aligned} \nu_2(\operatorname{ord}_q(b)) &= \nu_2\left(\frac{\varphi(q)}{\gcd(\varphi(q),r)}\right)\\ &= \nu_2(p^{\alpha-1}(p-1)) - \nu_2(\gcd(r,p^{\alpha-1}(p-1)))\\ &= \nu_2(p-1) - \nu_2(\gcd(r,p-1))\end{aligned}$$

Hence $\nu_2(\operatorname{ord}_q(b)) = \nu_2(p-1)$ is satisfied if and only if $r$ is odd and the claim follows from $\left(\frac{b}{p}\right) = \left(\frac{g^r}{p}\right) = (-1)^r$.

We will prove (iii) by induction over $\alpha$. The claim is of course satisfied for $\alpha = 1$. Now let $\alpha > 1$. Since $b^s \equiv 1 \pmod{p^\alpha}$ implies $b^s \equiv 1 \pmod{p^{\alpha-1}}$ it follows that $\operatorname{ord}_{p^{\alpha-1}}(b)$ divides $\operatorname{ord}_{p^\alpha}(b)$. Now let $t = \operatorname{ord}_{p^{\alpha-1}}(b)$. Thus $b^t = 1 + kp^{\alpha-1}$ for an integer $k$. If $k \equiv 0 \pmod{p}$ then it follows that $b^t \equiv 1 \pmod{p^\alpha}$ and thus $\operatorname{ord}_{p^\alpha}(b) = t$. If $k \not\equiv 0 \pmod{p}$ then it follows $(b^t)^n = (1+kp^{\alpha-1})^n \equiv 1 + nkp^{\alpha-1} \pmod{p^\alpha}$. Hence $(b^t)^n \equiv 1 \pmod{p^\alpha}$ is satisfied if and only if $p$ divides $n$ and thus $\operatorname{ord}_{p^\alpha}(b) = pt$. □

## 2.7 Lucas sequences

In this section we shall define a generalized version of the well-known Fibonacci sequence. Moreover, we shall show the connection between these sequences and quadratic number fields. We refer to [Bre89, chapter 12] for more details.

**Definition 2.11** *Let $P, Q$ be integers, $\Delta = P^2 - 4Q$ be not a square and $\alpha = \frac{P+\sqrt{\Delta}}{2}$. Then the Lucas sequences $V_k(P,Q)$ and $U_k(P,Q)$ are the rational numbers for which*

$$\alpha^k := \frac{V_k(P,Q)}{2} + \frac{U_k(P,Q)\sqrt{\Delta}}{2} \tag{2.15}$$

If $\mathbb{Q}(\sqrt{\Delta})$ is represented as a vector space of dimension 2 over $\mathbb{Q}$ with basis $\{1/2, \sqrt{\Delta}/2\}$ then $V_k(P,Q)$ and $U_k(P,Q)$ can be seen as the coefficients of $\alpha^k$. Since $\alpha$ is a root of $x^2 - Px + Q$ it follows that $\alpha \in \mathcal{O}_{\sqrt{\Delta}}$.

Let $\beta$ be the conjugate of $\alpha$ in $\mathbb{Q}(\sqrt{\Delta})$ i.e. $\beta = \frac{P-\sqrt{\Delta}}{2}$. Because conjugation and exponentiation are exchangeable we have

$$\beta^n = \overline{\alpha^n} = \frac{V_n(P,Q)}{2} - \frac{U_n(P,Q)\sqrt{\Delta}}{2}$$

From this equation and from (2.15) it follows that

$$V_n(P,Q) = \alpha^n + \beta^n \tag{2.16}$$

$$\text{and} \quad U_n(P,Q) = \frac{\alpha^n - \beta^n}{\alpha - \beta} \tag{2.17}$$

**Theorem 2.12** *Lucas functions satisfy the following properties.*

$$V_0(P,Q) = 2 \tag{2.18}$$
$$U_0(P,Q) = 0 \tag{2.19}$$
$$V_1(P,Q) = P \tag{2.20}$$
$$U_1(P,Q) = 1 \tag{2.21}$$
$$V_k(P,Q) = PV_{k-1}(P,Q) - QV_{k-2}(P,Q) \tag{2.22}$$
$$U_k(P,Q) = PU_{k-1}(P,Q) - QU_{k-2}(P,Q) \tag{2.23}$$
$$V_{-k}(P,Q) = Q^{-k}V_k(P,Q) \tag{2.24}$$
$$U_{-k}(P,Q) = -Q^{-k}U_k(P,Q) \tag{2.25}$$
$$4Q^k = V_k(P,Q)^2 - \Delta U_k(P,Q)^2 \tag{2.26}$$
$$V_{km}(P,Q) = V_k(V_m(P,Q), Q^m) \tag{2.27}$$
$$U_{km}(P,Q) = U_m(P,Q)U_k(V_m(P,Q), Q^m) \tag{2.28}$$
$$V_{k+m}(P,Q) = \frac{V_k(P,Q)V_m(P,Q)}{2} + \frac{\Delta U_k(P,Q)U_m(P,Q)}{2} \tag{2.29}$$
$$U_{k+m}(P,Q) = \frac{U_k(P,Q)V_m(P,Q)}{2} + \frac{V_k(P,Q)U_m(P,Q)}{2} \tag{2.30}$$

*Proof:* Many properties of Lucas functions can be derived by writing $\alpha^k$ in two different ways, replacing $\alpha^k$ by Lucas functions using (2.15) and comparing coefficients. Equations (2.18)–(2.21) follow from $\alpha^0 = 2/2 + 0\sqrt{\Delta}/2$ and $\alpha^1 = P/2 + 1\sqrt{\Delta}/2$. Moreover, we have $\alpha^2 = P\alpha - Q$ and therefore $\alpha^k = P\alpha^{k-1} - Q\alpha^{k-2}$. This proves equation (2.22) and (2.23). Equation (2.24) and (2.25) follow from

$$\begin{aligned} \frac{V_{-k}(P,Q)}{2} + \frac{U_{-k}(P,Q)\sqrt{\Delta}}{2} &= \alpha^{-k} = \left(\frac{\beta}{Q}\right)^k \\ &= Q^{-k}\left(\frac{V_{-k}(P,Q)}{2} - \frac{U_{-k}(P,Q)\sqrt{\Delta}}{2}\right) \end{aligned}$$

Equation (2.26) can be proved as follows.

$$\begin{aligned} 4Q^k &= 4(\alpha\overline{\alpha})^k = 2\alpha^k 2\overline{\alpha^k} \\ &= (V_k(P,Q) + U_k(P,Q)\sqrt{\Delta})(V_k(P,Q) - U_k(P,Q)\sqrt{\Delta}) \\ &= V_k(P,Q)^2 - \Delta U_k(P,Q)^2 \end{aligned}$$

Equation (2.26) now implies that

$$\begin{aligned} \alpha^k &= \frac{V_k(P,Q)}{2} + \frac{U_k(P,Q)\sqrt{\Delta}}{2} \\ &= \frac{V_k(P,Q)}{2} + \frac{\sqrt{U_k(P,Q)^2\Delta}}{2} \\ &= \frac{V_k(P,Q)}{2} + \frac{\sqrt{V_k(P,Q)^2 - 4Q^k}}{2} \end{aligned}$$

and hence $\alpha^k = P'/2 + \sqrt{P'^2 - 4Q'}/2$ with $P' = V_k(P,Q)$ and $Q' = Q^k$. Thus we have

$$\begin{aligned} (\alpha^k)^m &= \frac{V_m(P',Q')}{2} + \frac{U_m(P',Q')\sqrt{P'^2 - 4Q'}}{2} \\ &= \frac{V_m(P',Q')}{2} + \frac{U_m(P',Q')U_k(P,Q)\sqrt{\Delta}}{2}. \end{aligned}$$

Comparing the coefficients of this equation with $\alpha^{km} = V_{km}(P,Q)/2 + U_{km}(P,Q)\sqrt{\Delta}/2$ proves (2.27) and (2.28). Writing $\alpha^{k+m} = \alpha^k\alpha^m$ as sums of Lucas sequences and comparing the coefficients shows (2.29) and (2.30). $\square$

*Remark.* Lucas sequences are sometimes defined using the equations (2.18)–(2.23). We prefer (2.15) since by this the close relation to exponentiation cannot be overseen.

**Theorem 2.13** *For all positive integers $n$ and $P, Q \in \mathbb{Z}$ with $\gcd(Q,n) = 1$ there exists $t \in \mathbb{Z}$ such that $U_k(P,Q) \equiv 0 \pmod{n}$ if and only if $t$ divides $k$.*

*Proof:* Let $t$ be the smallest positive integer with $U_t(P,Q) \equiv 0 \pmod{n}$. It follows from (2.28) that $U_{rt}(P,Q) = U_t(P,Q)U_r(V_t(P,Q),Q^t) \equiv 0 \pmod{n}$ for all $r \in \mathbb{Z}$. Now assume that $U_k(P,Q) \equiv 0 \pmod{n}$. We can write $k = rt + s$ such that $r \in \mathbb{Z}$ and $0 \leq s < t$. Then we have

$$\begin{aligned} U_s(P,Q) &\equiv U_{k-rt}(P,Q) \\ &\equiv U_k(P,Q)V_{-rt}(P,Q) + U_{-rt}(P,Q)V_k(P,Q) \\ &\equiv Q^{-rt}U_k(P,Q)V_{rt}(P,Q) + Q^{-rt}U_{rt}(P,Q)V_k(P,Q) \\ &\equiv 0 \pmod{n} \end{aligned}$$

Note that the condition $\gcd(Q,n) = 1$ guarantees that $Q^{-rt} \bmod n$ is defined. Since $t$ is the smallest positive integer with $U_t(P,Q) \equiv 0 \pmod{p}$ it follows that $s = 0$. Thus $t$ divides $k$. □

**Theorem 2.14** *Let $p$ be an odd prime with $\gcd(\Delta,p) = 1$. Then*

$$\alpha^p \equiv \begin{cases} \alpha \pmod{p} & \textit{if} \quad \left(\frac{\Delta}{p}\right) = 1 \\ \overline{\alpha} \pmod{p} & \textit{if} \quad \left(\frac{\Delta}{p}\right) = -1 \end{cases} \tag{2.31}$$

*Proof:* Note that all computations are done in the ring of integers $\mathcal{O}_{\sqrt{\Delta}}$. We have $\alpha^p = (P/2 + \sqrt{\Delta}/2)^p \equiv P/2 + (\sqrt{\Delta})^p/2 \equiv P/2 + \Delta^{(p-1)/2}\sqrt{\Delta}/2 \equiv P/2 + \left(\frac{\Delta}{p}\right)\sqrt{\Delta}/2 \pmod{p}$. □

## 2.8 Dickson polynomials

Dickson polynomials have been introduced by L.Dickson [Dic97]. The Dickson polynomials of the first kind $D_n(x,a)$ are defined by

$$D_n(x,a) = \sum_{i=0}^{\lfloor n/2 \rfloor} \frac{n}{n-i} \binom{n-i}{i} (-a)^i x^{n-2i}$$

Dickson polynomials of the second kind $E_n(x,a)$ have been introduced by Schur [Sch73]. They are defined by

$$E_n(x,a) = \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n-i}{i} (-a)^i x^{n-2i}$$

An extended treatment of Dickson polynomials can be found in [LMT93]. Dickson polynomials and Lucas functions are essentially the same functions since

$$\begin{aligned} D_n(x,a) &= V_n(x,a) \\ E_n(x,a) &= U_{n+1}(x,a) \end{aligned}$$

## 2.9 The Rédei function

**Definition 2.15** *For $x \in \mathbb{Q}$, $a, n \in \mathbb{Z}$ with $a$ not being a square the Rédei function $f_n(x, a)$ is defined by*

$$f_n(x,a) = \sqrt{a}\frac{(x+\sqrt{a})^n + (x-\sqrt{a})^n}{(x+\sqrt{a})^n - (x-\sqrt{a})^n}. \tag{2.32}$$

This function was introduced and studied by Rédei [Réd48].

Equation (2.32) implies that $f_n(x,a) \in \mathbb{Q}(\sqrt{a})$. Moreover, it follows from $f_n(x,a) = \overline{f_n(x,a)}$ that $f_n(x,a) \in \mathbb{Q}$. The most important properties of Rédei functions are given in the sequel. A more complete treatment of Rédei functions can be found in [LMT93].

$$\left(\frac{x+\sqrt{a}}{x-\sqrt{a}}\right)^n = \frac{f_n(x,a)+\sqrt{a}}{f_n(x,a)-\sqrt{a}} \tag{2.33}$$

$$f_n(x,a) = \frac{V_n(2x, x^2-a)}{2U_n(2x, x^2-a)} \tag{2.34}$$

$$f_{n+m}(x,a) = \frac{f_n(x)f_m(x)+a}{f_n(x)+f_m(x)} \tag{2.35}$$

$$f_{nm}(x,a) = f_n(f_m(x)) \tag{2.36}$$

$$f_n\left(\frac{xy+a}{x+y}\right) = \frac{f_n(x,a)f_n(y,a)+a}{f_n(x,a)+f_n(y,a)} \tag{2.37}$$

*Proof:* (2.33) follows immediately from the definition. From (2.15) we have $(x+\sqrt{a})^n = V_n(2x,x^2-a)/2+U_n(2x,x^2-a)\sqrt{a}$ and by conjugation $(x-\sqrt{a})^n = V_n(2x,x^2-a)/2 - U_n(2x,x^2-a)\sqrt{a}$. If we substitute this into (2.32) then we get (2.34). Since $A = \frac{x+\sqrt{a}}{x-\sqrt{a}}$ has a unique solution $x = \sqrt{a}\frac{A+1}{A-1}$ for $x$ it follows that

$$\frac{x+\sqrt{a}}{x-\sqrt{a}} = \frac{y+\sqrt{a}}{y-\sqrt{a}}$$

implies $x = y$. Equation (2.35) thus follows from

$$\begin{aligned}&\left(\frac{f_m(x,a)+\sqrt{a}}{f_m(x,a)-\sqrt{a}}\right)\left(\frac{f_n(x,a)+\sqrt{a}}{f_n(x,a)-\sqrt{a}}\right)\\ = &\left(\frac{x+\sqrt{a}}{x-\sqrt{a}}\right)^{m+n} = \frac{f_{n+m}(x,a)+\sqrt{a}}{f_{n+m}(x,a)-\sqrt{a}}.\end{aligned}$$

Equation(2.36) can be shown in a similar way since

$$\begin{aligned}\frac{f_n(f_m(x,a),a)+\sqrt{a}}{f_n(f_m(x,a),a)-\sqrt{a}} &= \left(\frac{f_m(x,a)+\sqrt{a}}{f_m(x,a)-\sqrt{a}}\right)^n\\ = \left(\frac{x+\sqrt{a}}{x-\sqrt{a}}\right)^{mn} &= \frac{f_{nm}(x,a)+\sqrt{a}}{f_{nm}(x,a)-\sqrt{a}}\end{aligned}$$

Equation (2.37) follows from

$$\begin{aligned}
&\left(\frac{x+\sqrt{a}}{x-\sqrt{a}}\right)^n \left(\frac{y+\sqrt{a}}{y-\sqrt{a}}\right)^n \\
&= \left(\frac{f_n(x,a)+\sqrt{a}}{f_n(x,a)-\sqrt{a}}\right)\left(\frac{f_n(y,a)+\sqrt{a}}{f_n(y,a)-\sqrt{a}}\right) = \frac{\frac{f_n(x,a)f_n(y,a)+a}{f_n(x,a)+f_n(y,a)}+\sqrt{a}}{\frac{f_n(x,a)f_n(y,a)+a}{f_n(x,a)+f_n(y,a)}-\sqrt{a}}
\end{aligned}$$

and

$$\left(\frac{x+\sqrt{a}}{x-\sqrt{a}}\right)^n \left(\frac{y+\sqrt{a}}{y-\sqrt{a}}\right)^n = \left(\frac{\frac{xy+a}{x+y}+\sqrt{a}}{\frac{xy+a}{x+y}-\sqrt{a}}\right)^n = \frac{f_n\left(\frac{xy+a}{x+y},a\right)+\sqrt{a}}{f_n\left(\frac{xy+a}{x+y},a\right)-\sqrt{a}} \quad \square$$

If $p \neq 2$, $\left(\frac{a}{p}\right) = -1$ and $\gcd(n, p+1) = 1$ then $f_n(x,a)$ can be embedded into $\mathbb{F}_p$ using (2.34) by

$$f_n(x,a) \equiv V_n(2x, x^2-a)(2U_n(2x, x^2-a))^{-1} \pmod{p} \tag{2.38}$$

To show that this embedding is valid we have to show that $2U_n(2x, x^2-a) \not\equiv 0 \pmod{p}$. Let $\alpha = x + \sqrt{a}$ and $\Delta = a$. Theorem 2.14 shows that $\alpha^{p+1} \equiv \alpha\overline{\alpha} \equiv x^2 - a \pmod{p}$ is satisfied. Therefore we have $U_{p+1}(2x, x^2-a) \equiv 0 \pmod{p}$. If we assume that $U_n(2x, x^2-a) \equiv 0 \pmod{p}$ where $(n, p+1) = 1$ then it would follow by Theorem 2.13 that $U_1(2x, x^2-a) \equiv 0 \pmod{p}$. Because $a$ is not a quadratic residue module $p$ we have $U_1(2x, x^2-a) = x^2 - a \not\equiv 0 \pmod{p}$. Hence we have found a contradiction.
*Remark.* Rédei function have been used to design RSA-like cryptosystems [LM83], [Nöb85], [Wil85] and for pseudo-primality tests (see Section 3.2.6).

# Chapter 3

# Primality testing

## 3.1 Introduction

A large prime number can in principle be generated by repeatedly choosing an integer $n$ at random from the specified interval and testing $n$ for primality. The simplest primality test is to divide the given number $n$ by all primes less than or equal to $\sqrt{n}$, but this approach is completely infeasible when the length of $n$ exceeds 15–20 decimal digits. There exist several sophisticated general-purpose algorithms for testing primality [CL87],[Mor90] (see also [Len90]). Bosma and van der Hulst describe a very efficient version of the Jacobi sum test in [BvdH90]. This test has not a polynomial running time but it is nevertheless one of the fastest (if not the fastest) general-purpose primality test for practical sizes. F. Morain [Mor92] proved the primality of a 1505-digit number of a general form using an elliptic curve primality test in 1992. Even though the expected running time of this algorithm is polynomial in the average case it seems that it has a super-polynomial running time in the worst case. It should be pointed out that primality testing can be done in randomized polynomial time [AH92], but this algorithm is slower for the numbers that can currently be tested in reasonable time. Even larger numbers have been proven prime using tests that are suitable only for numbers of a special form. In January 1994 David Slowinski and Paul Gage proved that $2^{859433} - 1$ is prime. This is the largest known prime so far.

However, in practice *pseudo-primality tests* are commonly used. These kinds of tests return the correct result with high probability but not with certainty. Pseudo-primality tests are based on theorems of the form

> Every prime $n$ satisfies predicate $P$.

Thus if $P(n)$ is not satisfied then $n$ is composite. If $P(n)$ is satisfied for some composite numbers $n$ then the predicate $P$ cannot be used to prove primality. It is often the case that $P(n)$ is satisfied only for a small fraction of composite numbers. Thus observing that $P(n)$ is satisfied for $n$ may lead to the conjecture that $n$ is indeed prime. A number satisfying such a predicate will therefore be called a *probable prime*[1]. Composite numbers satisfying such a predicate $P(n)$ are called *pseudoprimes*. Different pseudo-primality tests and basic properties of these tests are described in Section 3.2.

Although the converse "$P(n)$ implies $n$ prime" is not generally true, it is sometimes possible to derive properties of the factors of $n$. Such theorems have the form

$P(n)$ implies $Q(p)$ for all [prime] factors $p$ of $n$.

When sufficiently many conditions on the prime factors of $n$ can be derived this can eventually lead to a proof that $n$ and 1 are the only possible factors of $n$ and hence that $n$ is prime. Section 3.3 summarizes results of this type. Another application of such conditions is to improve algorithms that generate some or all pseudoprimes up to a certain limit. In Section 3.5 algorithms for constructing counterexamples to pseudo-primality tests are presented and in Section 3.6 algorithms for finding all pseudoprimes up to a given limit are described. Such results will be used in Section 3.6.7 to prove the correctness[2] of certain pseudoprimality tests in given large intervals.

## 3.2 Pseudo-primality tests

### 3.2.1 The Fermat test

Fermat's (little) theorem states that

$$b^{n-1} \equiv 1 \pmod{n} \tag{3.1}$$

for every prime $n$ and base $b$ relatively prime to $n$. Thus if equation (3.1) is not satisfied for some $n$ and some $b$ relatively prime to $n$ then $n$ is composite. For example it follows from

$$3^{2^{2^6}} \equiv 8752249535465629170 \not\equiv 1 \pmod{2^{2^6}+1}$$

that the 6th Fermat number $2^{2^6}+1$ is composite. However, there exist composite numbers $n$ satisfying (3.1) for some $b$. For example $341 = 11 \cdot 31$ is composite and

$$2^{340} \equiv 1 \pmod{341}$$

It can be observed that such numbers are rare. There are only 101,629 Fermat pseudoprimes for the base 2 up to $10^{12}$ [Pin93c]. This number is very small

[1] Note that this is not an exact definition. Whether a number is a probable prime or not depends on our knowledge and reflects our conjecture on this number.

[2] in the sense that all probable primes are indeed prime numbers

compared to $\pi(10^{12}) \approx 3.7 \cdot 10^{10}$. Thus if a randomly chosen number satisfies (3.1) then it is very likely that this number is prime. The *Fermat test*, whose definition follows, is based on this observation.

**Fermat test**

Input: *Two integers* $n$,$b$ *with* $\gcd(n, b) = 1$.

Step 1: *If* $b^{n-1} \not\equiv 1 \pmod{n}$ *then return "*$n$ *is composite" else return "*$n$ *is a probable prime".*

We call a composite number satisfying (3.1) a *Fermat pseudoprime* for the base $b$. On the other hand any $b$ relatively prime to $n$ for which (3.1) is a witness with reference to the Fermat test for the compositeness of $n$.

We denote by $\mathbf{F}(n)$ the set of bases $b$ that satisfy (3.1) i.e.

$$\mathbf{F}(n) = \{b \in \mathbb{Z}/(n)^* : b^{n-1} \equiv 1 \pmod{n}\}$$

Since $b^n \equiv 1 \pmod{n}$ and $c^n \equiv 1 \pmod{n}$ implies $(bc)^n \equiv 1 \pmod{n}$ we have

**Theorem 3.1** $\mathbf{F}(n)$ *is a subgroup of* $\mathbb{Z}/(n)^*$.

Theorem 3.2 gives an exact expression for the cardinality of $\mathbf{F}(n)$. For a proof we refer to [BW80, Theorem 1] or [Mon80].

**Theorem 3.2** *For* $n = \prod_{i=1}^{r} p_i^{\alpha_i}$ *we have*

$$\#\mathbf{F}(n) = \prod_{i=1}^{r} \gcd(n-1, p_i - 1).$$

A number that is a Fermat pseudoprime for every base $b$ relative prime to $n$ is called a *Carmichael number*. Thus a Carmichael number is a composite number with no witness. Theorem 3.3 gives a characterization of Carmichael numbers [Car12].

**Theorem 3.3 (Carmichael)** *A Carmichael number is a squarefree number* $n$ *such that for every prime divisors* $p$ *of* $n$, $p-1$ *divides* $n-1$.

*Proof:* Let $n$ be squarefree and assume that for each prime $p$ dividing $n$, $p-1$ divides $n-1$. Then $k = (n-1)/(p-1)$ is an integer and for each $a$ with $\gcd(a, n) = 1$ we have $a^{n-1} = (a^{p-1})^k \equiv 1^k = 1 \pmod{p}$. Because this equation is true for any prime divisor $p$ of $n$ it follows from the Chinese Remainder Theorem that $a^{n-1} \equiv 1 \pmod{n}$. Therefore $n$ is a Carmichael number.

Now assume that $n$ is a Carmichael number and that $p^\alpha$ is a prime power dividing $n$. Let $g$ be a generator of the multiplicative group $\mathbb{Z}/(p^\alpha)^*$. Then it follows from $g^{n-1} \equiv 1 \pmod{n}$ that $g^{n-1} \equiv 1 \pmod{p^\alpha}$ and therefore that $|\mathbb{Z}/(p^\alpha)^*| = (p-1)p^{\alpha-1}$ divides $n-1$. Since $p$ cannot divide $n-1$ we have $\alpha = 1$ and also $(p-1)|(n-1)$. $\square$

The smallest Carmichael number is $561 = 3 \cdot 11 \cdot 17$. It is shown in [AGP94b] that there are infinitely many Carmichael numbers and that there are more than $x^{2/7}$ Carmichael numbers between 1 and $x$ for all sufficiently large $x$. Therefore the Fermat test should not be used as a pseudoprimality test. Tests that are more reliable will be presented in the following sections.

### 3.2.2 The Solovay-Strassen test

A stronger version of Fermat's Theorem is the following: If $n$ is an odd prime and $b$ is relatively prime to $n$ then

$$b^{(n-1)/2} \equiv \left(\frac{b}{n}\right) \pmod n \tag{3.2}$$

The *Solovay-Strassen test* is based on this equation.

**Solovay Strassen**

Input: *Two integers* $n$ *and* $b$ *such that* $\gcd(n, b) = 1$
Step 1: *If* $b^{(n-1)/2} \not\equiv \left(\frac{b}{n}\right) \pmod n$ *then return "$n$ is composite" else return "$n$ is a probable prime".*

Like for the Fermat test compositeness of a number $n$ can be proved with certainty, but a proof of primality cannot be obtained from the Solovay-Strassen test. For example (3.2) is satisfied for $n = 561$ and $b = 2$. We call a composite $n$ *Euler pseudoprime* for the base $b$ if (3.2) is satisfied. Let $\mathbf{E}(n)$ denote the set of bases satisfying (3.2) i.e.

$$\mathbf{E}(n) = \{b \in \mathbb{Z}/(n)^* : b^{(n-1)/2} \equiv \left(\frac{b}{n}\right) \pmod n\}$$

Since $b^{(n-1)/2} \equiv \left(\frac{b}{n}\right) \pmod n$ implies $b^{n-1} \equiv \left(\frac{b}{n}\right)^2 \equiv 1 \pmod n$ we see that the Solovay-Strassen test implies the Fermat test and in particular we have shown Theorem 3.4.

**Theorem 3.4** $\mathbf{E}(n)$ *is a subset of* $\mathbf{F}(n)$.

The Solovay-Strassen test can recognize composite numbers that cannot be recognized as composite by the Fermat test. For example 561 is a Carmichael number but

$$5^{280} \equiv 67 \not\equiv \left(\frac{5}{561}\right) \pmod{561}$$

shows that 561 is not a Euler pseudoprime for the base 5. Furthermore Theorem 3.5 shows that no composite number can be an Euler pseudoprime for all bases.

**Theorem 3.5** *Let $n$ be an odd composite. Then $\mathbf{E}(n)$ is a proper subgroup of $\mathbb{Z}/(n)^*$.*

*Proof:* It follows from $(ab)^n = a^n b^n \equiv \left(\frac{a}{n}\right)\left(\frac{b}{n}\right) = \left(\frac{ab}{n}\right) \pmod{n}$ that $\mathbf{E}(n)$ is a subgroup of $\mathbb{Z}/(n)^*$. Thus it remains to show that the assumption $\mathbf{E}(n) = \mathbb{Z}/(n)^*$ leads to a contradiction. It follows from $\mathbf{E}(n) = \mathbb{Z}/(n)^*$ that $n$ is a Carmichael number and therefore squarefree. Let $p$ be a prime divisors of $n$, $m = n/p$ and let $c_0$ be an integer satisfying $\left(\frac{c_0}{p}\right) = -1$. Since $\gcd(m,p) = 1$ it follows from the CRT that there exists $c$ with $c \equiv c_0 \pmod{p}$ and $c \equiv 1 \pmod{m}$. We have $\left(\frac{c}{n}\right) = \left(\frac{c}{m}\right)\left(\frac{c}{p}\right) = -1$. Thus $c^{(n-1)/2} \equiv -1 \pmod{n}$ and therefore $-1 \equiv c^{(n-1)/2} \equiv 1^{(n-1)/2} \equiv 1 \pmod{m}$, which is a contradiction. □

Because the cardinality of a subgroup divides the cardinality of the group we have

**Corollary 3.6** *$\#\mathbf{E}(n) \le \frac{1}{2}\varphi(n)$ for all odd composite $n$.*

An exact formula for the cardinality of $\mathbf{E}(n)$ has been given by Monier in [Mon80].

**Theorem 3.7** *Let $n > 1$ be odd. Then*

$$\#\mathbf{E}(n) = \delta(n) \prod_{p|n} \gcd(\frac{n-1}{2}, p-1)$$

*where*

$$\delta(n) = \begin{cases} 2 & \text{if } \nu_2(p-1) \ge \nu_2(n-1) \text{ for all } p|n \\ 1/2 & \text{if } \exists\, p|n \text{ with } \nu_2(p-1) < \nu_2(n-1) \text{ and } \nu_p(n) \text{ odd} \\ 1 & \text{otherwise} \end{cases}$$

### 3.2.3 The Miller-Rabin test

This test is based on the fact that the equation

$$x^2 \equiv 1 \pmod{n}$$

has only two solutions $x = \pm 1 \pmod{n}$ if $n$ is prime but more solutions if $n$ is composite. Let $n = u2^v + 1$ be an odd prime where $u$ is odd. It follows that

$$b^u \bmod n,\ b^{2u} \bmod n,\ b^{2^2 u} \bmod n,\ \ldots,\ b^{2^v u} \bmod n$$

must not contain two successive numbers $b^{2^i} u \bmod n$ and $b^{2^{i+1} u} \bmod n$ such that $b^{2^{i+1} u} \equiv 1 \pmod{n}$ and $b^{2^i u} \not\equiv \pm 1 \pmod{n}$ if $n$ is prime, since this would imply that a square root of 1 different from $\pm 1$ exists. Thus if $n = u2^v + 1$ is an odd prime and $u$ is odd then either

$$b^u \equiv 1 \pmod{n} \tag{3.3}$$

or

$$b^{u2^c} \equiv -1 \pmod{n} \tag{3.4}$$

with $0 \le c < v$ is satisfied.

**Miller-Rabin test for base $b$**

Input: *Two integers $n$ and $b$ such that* $\gcd(n, b) = 1$.

Step 1: *Set $v \leftarrow \nu_2(n-1)$ and $u \leftarrow (n-1)/2^v$.*

Step 2: *Set $x_0 \leftarrow b^u \pmod{n}$. If $x_0 \equiv 1 \pmod{n}$ then return "$n$ is a probable prime" otherwise compute iteratively $x_i \equiv x_{i-1}^2 \pmod{n}$ for $1 \le i < v$. If $x_i \equiv -1$ for $0 \le i < v$ then return "$n$ is a probable prime". Otherwise return "$n$ is composite".*

A composite $n$ is called a *strong pseudoprime* for the base $b$ if the Miller-Rabin test returns "$n$ is a probable prime". For $n = u2^v + 1$ with $u$ odd we denote by

$$\mathbf{S}(n) = \{b \in \mathbb{Z}/(n)^* : b^u \equiv 1 \pmod{n} \text{ or } b^{u2^c} \equiv -1 \pmod{n} \text{ for } 0 \le c < v\}$$

the set of bases for which $n$ is a strong pseudoprime.

**Theorem 3.8** *If $n > 1$ is an odd composite number then*

*(i)* $\mathbf{S}(n)$ *is a subset of* $\mathbf{E}(n)$.

*(ii) If* $n \equiv 3 \pmod{4}$ *then* $\mathbf{S}(n) = \mathbf{E}(n)$.

*(iii)* $\#\mathbf{S}(n) \le \varphi(n)/4$.

*(iv) If $u$ is the largest odd divisor of $n-1$ and $\nu(n) = \min_{p|n}\{\nu_2(p-1)\}$ and $\omega(n)$ is the number of prime factors of $n$ then*

$$\#\mathbf{S}(n) = \left(1 + \frac{2^{\nu(n)\omega(n)} - 1}{2^{\omega(n)-1}}\right) \prod_{p|n} \gcd(u, p-1)$$

For proofs we refer to [PSW80, Theorem 3] for (i), [Mal77] for (ii) and [Mon80] for (iii) and (iv).

**Definition 3.9** *Let $b$ and $m = \prod_{i=1}^{s} {p_i}^{a_i}$ be to integers that are relatively prime to each other. Then $m$ is called* $b$-admissible *if there exists $c$ such that $\nu_2(\mathrm{ord}_{{p_i}^{a_i}}(b)) = c$ for all $1 \le i \le s$.*

Note that by Theorem 2.10 (iii) we have $\nu_2(\mathrm{ord}_{p^a}(b)) = \nu_2(\mathrm{ord}_p(b))$ for all odd primes $p$. An odd integer $m$ is therefore $b$-admissible if and only if $\nu_2(\mathrm{ord}_p(b))$ takes the same value for all prime divisors $p$ of $m$. Moreover, if $m$ is $b$-admissible then $\nu_2(\mathrm{ord}_{p_i}(b)) = \nu_2(\mathrm{ord}_m(b))$.

The following theorem gives necessary and sufficient conditions for a Fermat pseudoprime to be also a *strong* pseudoprime. Jaeschke [Jae93, Proposition 1] gives a proof of a version of this theorem that is restricted to squarefree integers.

**Theorem 3.10** *An odd Fermat pseudoprime $m$ is a strong pseudoprime for the base $b$ if and only if $m$ is $b$-admissible.*

*Proof:* Let $m = \prod_{i=1}^{s} {p_i}^{a_i} = u2^v + 1$ with $u$ odd. We have $b^{u2^v} \equiv 1 \pmod{m}$ because $m$ is a Fermat pseudoprime for the base $b$.

Now assume that $m$ is $b$-admissible. Thus there exists $c$ such that $\nu_2(\mathrm{ord}_{{p_i}^{a_i}}(b)) = c$ for all $1 \le i \le s$ and therefore $\mathrm{ord}_{{p_i}^{a_i}}(b) = u_i 2^c$ is satisfied for some odd integer $u_i$. Moreover, it follows from $b^{u2^v} \equiv 1 \pmod{p_i^{a_i}}$ that $u_i 2^c$ divides $u2^v$. Thus $u_i$ divides $u$ and hence $b^{u2^c} \equiv 1 \pmod{p_i^{a_i}}$. This equation is true for all $1 \le i \le s$, so it follows from the CRT that $b^{u2^c} \equiv 1 \pmod{m}$.

Moreover, if $c > 0$ then $\mathrm{ord}_{{p_i}^{a_i}}(b)$ does not divide $u2^{c-1}$ and so $b^{u2^{c-1}} \not\equiv 1 \pmod{p_i^{a_i}}$. Because $(b^{u2^{c-1}})^2 \equiv b^{u2^c} \equiv 1 \pmod{p_i^{a_i}}$ it follows that $b^{u2^{c-1}} \equiv -1 \pmod{p_i^{a_i}}$ and by CRT that $b^{u2^{c-1}} \equiv -1 \pmod{m}$. Thus $m$ is a strong pseudoprime for the base $m$.

Conversely, assume $m$ is a strong pseudoprime. Either $b^u \equiv 1 \pmod{m}$ or $b^{u2^d} \equiv -1 \pmod{m}$ for some $0 \le d < v$. In the former case we have $b^u \equiv 1 \pmod{{p_i}^{a_i}}$ for $1 \le i \le s$ and thus $\mathrm{ord}_{{p_i}^{a_i}}(b)$ divides $u$. Hence $\nu_2(\mathrm{ord}_{p_i^{a_i}}(b)) = 0$ for all $1 \le i \le s$.

In the latter case we have $b^{u2^d} \equiv -1 \pmod{p_i^{a_i}}$ for all $1 \le i \le s$. Thus $\mathrm{ord}_{p_i^{a_i}}$ divides $u2^{d+1}$ but does not divide $u2^d$. It follows that $\nu_2(\mathrm{ord}_{p_i^{a_i}}) = d+1$ for all $1 \le i \le s$. So $m$ is $b$-admissible. $\square$

### 3.2.4 Lucas-based tests

Let $P$ and $Q$ be two integers such that $\Delta = P^2 - 4Q$ is not a perfect square and let $\alpha = \frac{P+\sqrt{\Delta}}{2}$. Theorem 2.14 asserts that (3.5) is satisfied for any prime $n$ with $\gcd(n, \Delta) = 1$ in $\mathcal{O}_{\sqrt{\Delta}}$.

$$\alpha^n \equiv \begin{cases} \alpha \pmod{n} & \text{if } \left(\frac{\Delta}{n}\right) = 1 \\ \overline{\alpha} \pmod{n} & \text{if } \left(\frac{\Delta}{n}\right) = -1 \end{cases} \tag{3.5}$$

We refer to a composite number $n$ that satisfies (3.5) as a quadratic field-based pseudoprime[3] (*QF pseudoprime*) for the base $\alpha$. Alternatively we call $n$ a QF pseudoprime for the parameters $P, Q$ whenever this choice is more convenient.

#### QF test

Input: *Integers $n$,$P$, $Q$ such that* $\gcd(n, P^2 - 4Q) = 1$.

Step 1: *Set* $\Delta \leftarrow P^2 - 4Q \bmod n$ *and* $\alpha \leftarrow \frac{P}{2} + \frac{\sqrt{\Delta}}{2}$.

Step 2: *Compute* $z \leftarrow \alpha^n \bmod n$.

[3]Note that we introduce this new term here in order not to confuse with the existing terms Lucas pseudoprime and Fibonacci pseudoprime. The concept of QF pseudoprimes, however, is not original.

Step 3: *If* $\left(\frac{\Delta}{n}\right) = 1$ *and* $z \not\equiv \alpha \pmod{n}$ *or* $\left(\frac{\Delta}{n}\right) = -1$ *and* $z \not\equiv \overline{\alpha} \pmod{n}$ *then return "n is composite". Otherwise return "n is a probable prime".*

This test can be defined in terms of Lucas sequences. It follows immediately from equation (2.15) that (3.5) is equivalent to the combination of

$$V_n(P,Q) \equiv P \pmod{n} \tag{3.6}$$

$$\text{and} \quad U_n(P,Q) \equiv \left(\frac{\Delta}{n}\right) \pmod{n} \tag{3.7}$$

If $\gcd(n,Q) = 1$ then it follows from $\alpha\overline{\alpha} = Q$ that $\alpha^{-1} \equiv Q^{-1}\overline{\alpha} \pmod{n}$ and that $\alpha$ is invertible in $\mathcal{O}_{\sqrt{\Delta}}$. Thus if $\gcd(n,Q) = 1$ then (3.5) is equivalent to

$$\alpha^{n-1} \equiv 1 \pmod{n} \quad \text{if} \quad \left(\frac{\Delta}{n}\right) = 1 \tag{3.8}$$

$$\alpha^{n+1} \equiv Q \pmod{n} \quad \text{if} \quad \left(\frac{\Delta}{n}\right) = -1 \tag{3.9}$$

Equations (3.8) and (3.9) can be expressed using Lucas sequences. If $\epsilon = \left(\frac{\Delta}{n}\right)$ and $\gcd(n,Q) = 1$ then (3.5) implies

$$U_{n-\epsilon}(P,Q) \equiv 0 \pmod{p} \tag{3.10}$$

$$V_{n-\epsilon}(P,Q) \equiv \begin{cases} 2 \text{ if } \left(\frac{\Delta}{n}\right) = 1 \\ 2Q \text{ if } \left(\frac{\Delta}{n}\right) = -1 \end{cases} \tag{3.11}$$

Other tests based on Lucas sequences have been studied. A *Lucas pseudoprime* is a composite number $n$ that satisfies only (3.10)[BW80]. A *Fibonacci pseudoprime* of the $P$-th kind is a composite number $n$ that satisfies (3.6) for $Q = -1$ [PF88]. Pinch [Pin93a] has found a composite number $n$ that is a Fibonacci pseudoprime of the $P - th$ kind for all $P$ relatively prime to $n$.

It should be noted that $U_n(P,Q)$ is normally computed together with $V_n(P,Q)$. Testing both, (3.10) and (3.11) is therefore almost as efficient as testing only one of these equations.

**Theorem 3.11** *If $n$ is a QF pseudoprime for the base $\alpha$ then $n$ is a Fermat pseudoprime for the base $Q = \alpha\overline{\alpha}$.*

*Proof:* Exponentiation and conjugation are exchangeable. Therefore we have $Q^n \equiv (\alpha\overline{\alpha})^n \equiv \alpha^n\overline{\alpha^n} \equiv \alpha\overline{\alpha} \equiv Q \pmod{n}$. □

Conversely assume that a number $n$ is a Fermat pseudoprime for two bases $b$ and $c$. It follows from (2.16) that $V_n(b+c, bc) = b^n + c^n \equiv b + c \pmod{n}$ and by (2.17) that $U_n(b+c, bc) = (b^n - c^n)/(b-c) \equiv (b-c)/(b-c) \equiv 1 \pmod{n}$. Thus $n$ is a QF pseudoprime for the parameters $P = b+c$ and $Q = bc$. Therefore if $n$

is a Carmichael number then $n$ is a QF pseudoprime for all $P$ and $Q$ for which $P^2 - 4Q$ is a quadratic residue modulo $n$.

If a pseudo-primality test is a combination of a Fermat test (or Miller-Rabin test) and a QF test then one would like these tests to be independent. This is not the case, when $P^2 - 4Q$ is a quadratic residue. Therefore it seems reasonable to choose $P$ and $Q$ such that $\left(\frac{P^2-4Q}{n}\right) = -1$.

The QF test implies the Solovay-Strassen test. In particular we have

**Theorem 3.12** *Let $n$ be an odd composite integer and let $Q$ be relatively prime to $n$. Then $n$ is an Euler pseudoprime for the base $Q$ if and only if it is a QF pseudoprime for the base $\sqrt{Q}$.*

*Proof:* Since $\overline{\sqrt{Q}} = -\sqrt{Q}$ it follows that the QF test is equivalent to $\sqrt{Q}^n \equiv \left(\frac{Q}{n}\right)\sqrt{Q} \pmod{n}$. Since $Q$ is relatively prime to $n$ the preceding equation is equivalent to $Q^{(n-1)/2} \equiv \left(\frac{Q}{n}\right) \pmod{n}$. $\square$

*Remark.* Even though Theorem 3.12 implies that the the QF test is as strong as the Euler test it should be noted that chosing $\sqrt{Q}$ as base for the QF test is rather bad since it turns out that the QF test is generally much more reliable than the Solovay-Strassen test. Algorithms for chosing better bases are described in Section 3.4.

### 3.2.5 Extended versions of the QF test

In this section we will consider generalizations of the QF test that are similar to the generalizations of the Fermat test has been extended to the Solovay-Strassen and Miller-Rabin test.

**Lemma 3.13** *Let $x \in \mathcal{O}_{\sqrt{\Delta}}$ and let $n$ be an odd prime with $\gcd(n, \Delta) = 1$. If*

$$x^2 \equiv t \pmod{n}$$

*for some $t \in \mathbb{Z}$ and $\gcd(t, n) = 1$ then either $x \in \mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$ or $x \in \sqrt{\Delta}\mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$. In particular if $\left(\frac{\Delta}{n}\right) = -1$ then $x \in \mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$ if $\left(\frac{t}{n}\right) = 1$ and $x \in \sqrt{\Delta}\mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$ if $\left(\frac{t}{n}\right) = -1$.*

*Proof:* Assume that $x \equiv r + s\sqrt{\Delta} \pmod{n}$. From $(r + s\sqrt{\Delta})^2 = r^2 + \Delta s^2 + 2rs\sqrt{\Delta} \equiv t \pmod{n}$ follows $2rs \equiv 0 \pmod{n}$ and thus $r \equiv 0 \pmod{n}$ or $s \equiv 0 \pmod{n}$. This implies the first part of the Lemma.

Now assume that $\left(\frac{\Delta}{n}\right) = -1$. Then $\mathcal{O}_{\sqrt{\Delta}}/(n)$ is a field of characteristic $n \neq 2$. Therefore $x^2 = t$ has at most two roots in $\mathcal{O}_{\sqrt{\Delta}}/(n)$. $\left(\frac{t}{n}\right) = 1$ implies that the roots of $x^2 \equiv t \pmod{n}$ are indeed in $\mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$. If $\left(\frac{t}{n}\right) = -1$ then $x'^2 \equiv t\Delta^{-1} \pmod{n}$ has two solutions in $\mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$ and thus $(x'\sqrt{\Delta})^2 \equiv t \pmod{n}$ shows the second part of the Lemma. $\square$

*Remark.* Note that $\mathcal{O}_{\sqrt{\Delta}}/(n)$ is not a field when $\left(\frac{\Delta}{n}\right) = 1$ and that $x^2 = t$ may have more than two solutions for $x$. (e.g. in $x^2 = 1$ has four solutions $x = 1, 10, 3\sqrt{5}, 8\sqrt{5}$ in $\mathcal{O}_{\sqrt{5}}/(11)$)

Lemma 3.13 can be used to define improved versions of the QF test. Let $\epsilon = \left(\frac{\Delta}{n}\right)$. We shall call an odd composite $n$ a *Euler QF pseudoprime* for the base $\alpha$ if $\alpha$ is a *QF pseudoprime* and

$$\alpha^{(n-\epsilon)/2} \in \mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}} \text{ or } \sqrt{\Delta}\mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$$

If $n - \epsilon = u2^v$ with $u$ odd then we may consider the sequence

$$\alpha^u \bmod n, \quad \alpha^{2u} \bmod n, \quad \alpha^{2^2 u} \bmod n, \ldots, \alpha^{2^v u} \bmod n$$

If $n$ is prime then it follows from Lemma 3.13 and $\alpha^{n-\epsilon} \in \mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$ that either $\alpha^u \in \mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$ or $\alpha^{2^i u} \in \sqrt{\Delta}\mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$ for some $0 \le i < v$. A QF pseudoprime $n$ satisfying this condition is called a *strong QF pseudoprime.* Note that these definitions are analogous to the definition of Euler Lucas pseudoprimes and strong Lucas pseudoprimes in [BW80].

An example showing a QF pseudoprimes which is not an Euler QF pseudoprimes or a strong QF pseudoprimes is $n = 365, P = 1, Q = 147, \Delta = P^2 - 4Q = -587$ because

$$\left(\frac{1}{2} + \frac{\sqrt{-587}}{2}\right)^{365+1} \equiv 147 \pmod{n}$$

but

$$\left(\frac{1}{2} + \frac{\sqrt{-587}}{2}\right)^{(365+1)/2} \equiv 145 + 73\sqrt{-587} \pmod{n}.$$

**Strong QF test for base $\alpha$**

Input: *Integers $n$ and $P, Q$ such that* $\gcd(n, Q) = \gcd(n, P^2 - 4Q) = 1$.

Step 1: *Set $\Delta = P^2 - 4Q$, $\epsilon = \left(\frac{\Delta}{n}\right)$, $v \leftarrow \nu_2(n - \epsilon)$ and $u \leftarrow (n - \epsilon)/2^v$.*

Step 2: *Set $x_0 \leftarrow \alpha^u \bmod n$. If $x_0 \in \mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$ then set $b \leftarrow$ TRUE else $b \leftarrow$ FALSE.*

Step 3: *For $i$ from $0$ to $v - 1$ do: If $x_i \in \sqrt{\Delta}\mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$ then $b \leftarrow$ TRUE. Now set $x_{i+1} \leftarrow x_i^2 \bmod n$.*

Step 4: *If $\left(\frac{\Delta}{n}\right) = 1$ then set $y \leftarrow \alpha$ otherwise set $y \leftarrow \overline{\alpha}$.*

Step 5: *If $x_v \equiv y \pmod{n}$ and $b =$ TRUE then return "$n$ is a probable prime" otherwise return "$n$ is composite".*

*Remarks.* (1) Pinch observed that the function `PrimeQ` of Mathematica 2.2 returns TRUE for $n = 89 \cdot 11551 \cdot 37159$ [Pin93d]. This failure was caused by a weak version of the Lucas test. This test has been replaced by the preceding version of the QF test with the parameters $P \equiv a^{-1} \pmod{n}$ and $Q = 1$ where $a$ is the smallest positive integer satisfying $\left(\frac{1-4a^2}{n}\right) = -1$.

(2) If $\left(\frac{\Delta}{n}\right) = -1$ then the second part of Lemma 3.13 specifies the set $(\mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$ or $\sqrt{\Delta}\mathbb{Z} + \mathcal{O}_{\sqrt{\Delta}})$ in which a square root of an integer lies if $n$ is a prime. Testing for the correct set does not lead to an improved version of a test since any Euler QF pseudoprime satisfies the same condition. If $\alpha^{(n+1)/2} \in \mathbb{Z} + \mathcal{O}_{\sqrt{\Delta}}$ then $\alpha^{n+1} \equiv Q \pmod{n}$ is a quadratic residue $\mathbb{Z}/(n)$ and if $\alpha^{(n+1)/2} \in \sqrt{\Delta}\mathbb{Z} + n\mathcal{O}_{\sqrt{\Delta}}$ then $\alpha^{n+1} \equiv t^2\Delta \equiv Q \pmod{p}$ for some $t \in \mathbb{Z}$ and therefore it follows that $\left(\frac{Q}{n}\right) = -1$.

(3) Davenport has proposed to count the number of different square roots of $-1 \bmod n$ that appear in different passes of the Miller Rabin test. If more than two square roots can be found then $n$ is composite [Dav92]. The same idea can be applied to the QF test if $Q$ is fixed and different values for $P$ with $\left(\frac{\Delta}{n}\right) = -1$ are tried. Note that it is possible to apply this test to any odd $n$ whereas Davenport's refinement for the Miller-Rabin test is possible only when $n \equiv 1 \pmod{4}$.

(4) Another idea is to chose $P$, $Q$, $\Delta$ with $\left(\frac{\Delta}{n}\right) = -1$ in such a way that a square root $y$ of $Q$ is known. Now if $\alpha^{(n+1)/2} \not\equiv \pm y \pmod{n}$ then $n$ is composite. One possibility to proceed is to choose $t, P \in \mathbb{Z}$ such that $\left(\frac{1+4t^2}{n}\right) = -1$ and to set $\Delta \leftarrow \frac{P^2}{1+4t^2} \bmod n$, $Q \leftarrow t^2\Delta$ and $y \leftarrow t\sqrt{\Delta}$.

### 3.2.6 Tests based on Rédei functions

Another test based on the Rédei function $f_n(x)$ (see Definition 2.15) has been proposed by Lidl and Müller [LM91]. They observe that

$$b^{(n-1)/2} f_n(x, b) \equiv x \pmod{n} \tag{3.12}$$

if $n$ is a prime and $\left(\frac{b}{n}\right) = -1$. A composite number satisfying (3.12) is called a *Rédei pseudoprime* of kind $(b, x)$. Assume that a composite $n$ is an Euler pseudoprime for the base $b$ with $\left(\frac{b}{n}\right)$ and a QF pseudoprime for the base $\alpha = P/2 + \sqrt{\Delta}/2$ where $P = 2x$, $Q = x^2 - b$ and $\Delta = P^2 - 4Q = 4b$. Then it follows that $\left(\frac{\Delta}{n}\right) = \left(\frac{b}{n}\right)$ and

$$\begin{aligned} b^{(n-1)/2} f_n(x, b) &= b^{(n-1)/2} \frac{V_n(P, Q)}{2U_n(P, Q)} \\ &= \left(\frac{b}{n}\right) \frac{P}{2\left(\frac{\Delta}{n}\right)} \\ &= P/2 = x \end{aligned}$$

Thus $n$ is also a Rédei pseudoprime of kind $(x, b)$. However, the converse is generally not true. An Euler and a QF test should therefore be performed instead of a Rédei pseudoprime test, unless there is a more efficient way to compute $b^{(n-1)/2} f_n(x,b)$ than to compute $b^{(n-1)/2}$ and $(x+\sqrt{a})^n$ separately.

## 3.3 Conditions on factors of pseudoprimes

The Fermat test can be used to prove efficiently compositeness of a number $n$. Unfortunately, as already noted, it cannot directly be used to prove primality. However, if it is modified appropriately it can be used to find properties of all divisors of $n$. If sufficiently many such properties are known this can eventually lead to a proof of primality of $n$. A theorem that can be used towards this goal is the following

**Theorem 3.14** *Let $n$ be an integer, $p$ be a prime number dividing $n-1$ and $b$ be relatively prime to $n$. If*

$$\begin{aligned} b^{n-1} &\equiv 1 \pmod{n} && (3.13)\\ \gcd(b^{(n-1)/p}-1, n) &= 1 && (3.14) \end{aligned}$$

*then every divisor $d$ of $n$ satisfies*

$$d \equiv 1 \pmod{p^{\nu_p(n-1)}} \qquad (3.15)$$

*Proof:* Note that it is sufficient to show equation (3.15) for $d$ prime. Equation (3.13) implies that $\mathrm{ord}_d(b)$ divides $n-1$ and (3.14) implies that $\mathrm{ord}_d(b)$ does not divide $(n-1)/p$. Therefore $p^{\nu_p(n-1)}$ is a divisor of $\mathrm{ord}_d(b)$ and by Theorem 2.9 a divisor of $d-1$. □

Theorem 3.14 can be used to derive the following theorem by Pocklington [Poc16]

**Theorem 3.15 (Pocklington)** *Let $n = RF+1$ and $F = \prod_{i=1}^{s} p_i^{a_i}$. If there exists a base $b$ with*

$$\begin{aligned} b^{n-1} &\equiv 1 \pmod{n}\\ \gcd(b^{(n-1)/p_i}-1, n) &= 1 \text{ for all } 1 \le i \le s \end{aligned}$$

*then every divisor $d$ of $n$ satisfies $d \equiv 1 \pmod{F}$. Moreover, if $F > \sqrt{n}$ then $n$ is prime.*

*Proof:* It follows from Theorem 3.14 that $d \equiv 1 \pmod{p_i^{a_i}}$ for all $1 \le i \le s$ and by the CRT $d \equiv 1 \pmod{F}$. Now if $F \ge \sqrt{n}$ then every factor of $n$ except 1 is greater than $\sqrt{n}$ and thus $n$ cannot be composite. □

*Remarks.*

(1) An integer $n$ can be proved prime with Pocklington's theorem only if sufficiently many factors of $n-1$ are known. Factoring $n-1$ is, however, not always feasible when $n$ is large.

(2) It is possible to improve this theorem. If $F \geq \sqrt[3]{n}$ then a method is known to derive all factors of $n$ [Len84].

(3) Pocklington's theorem is very useful for generating primes since it is possible to generate possible prime numbers $n$ in such a way that sufficiently many prime factors of $n-1$ are known [Mau95].

## 3.4 Deterministic versions of pseudo-primality tests

In the Section 3.2 we have not specified how to choose the bases for the pseudo-primality tests. There are two fundamentally different approaches for this task. The first approach is to select randomly a base among all possible bases. Using this method it is for example possible to design for every $\epsilon > 0$ a test such that the probability of a composite number passing this test as "probable prime" is less than $\epsilon$. The second approach is to select a base (or a set of bases) by using a deterministic algorithm. Then any composite $n$ either is always recognized as composite or passes as probable prime. This has the advantage that it is possible to find intervals in which a pseudo-primality test recognizes all composite numbers. The test can therefore be used as a primality test in this interval. The goal of this section is to describe methods to find good bases or, more precisely, to avoid bad bases. Most of what follows is based on heuristic arguments.

### 3.4.1 Miller-Rabin test

A necessary condition for a composite number $n$ to pass the Miller-Rabin test for the base $b$ is that for all divisors $p^\alpha$ of $n$, $\mathrm{ord}_{p^\alpha}(b)$ divides $n-1$. Thus we would get a strong test if we could select a base $b$ for which $\mathrm{ord}_{p^a}(b)$ is large. Since the divisors of $n$ are unknown it seems possible to avoid only some bad bases. First $b \equiv \pm 1 \pmod{p}$ should be avoided. If $n$ is a strong pseudoprime for the base $b$ then $n$ is also a strong pseudoprime for the bases $b^k : k \in \mathbb{Z}$. Therefore powers of integers should not be chosen.

If the Miller-Rabin test is performed with more than one base then the bases should not be dependent. If $n$ is a strong pseudoprime for the bases $a$ and $b$ then it is very likely that $n$ is a strong pseudoprime for the base $ab$, since $n$ is at least an Euler pseudoprime for the base $ab$. Thus performing the Miller-Rabin test with the bases $a, b$ and $ab$ is not much stronger than an Miller-Rabin test with only $a$ and $b$. It seems however reasonable to choose a set of small primes as bases for the Miller-Rabin test.

| $n$ | bases $(P,Q)$ |
|---|---|
| 27 | $(\pm 5,-1)$ |
| 35 | $(\pm 1,1),(\pm 6,1)$ |
| 51 | $(\pm 7,-1),(\pm 10,-1),(\pm 24,16),(\pm 17,16)$ |
| 63 | $(\pm 18,1)$ |
| 65 | $(\pm 1,1),(\pm 14,1),(\pm 12,14),(\pm 27,14),$ $(\pm 31,-14),(\pm 21,-14),(\pm 8,-1),(\pm 18,-1)$ |
| 75 | $(\pm 25,26)$ |
| 77 | $(\pm 1,1),(\pm 34,1)$ |

Table 3.1: The QF pseudoprimes $n < 80$.

### 3.4.2 QF test

In the case of the QF test it is advisable to choose $P$ and $Q$ such that $\left(\frac{\Delta}{n}\right) = -1$ for $\Delta = P^2 - 4Q$. Now assume that $\left(\frac{\Delta}{p}\right) = -1$ for a prime divisor $p$ of $n$. Let $\alpha = P/2 + \sqrt{\Delta}/2$. It follows from $a^p \equiv \overline{\alpha} \pmod{p}$ that $\alpha^i \equiv \overline{\alpha} \pmod{p}$ if and only if $i \equiv p \pmod{\mathrm{ord}_p\alpha}$. Therefore we would like to choose $P, Q$ such that $\mathrm{ord}_p\alpha$ is large for all primes $p$ dividing $n$. A very bad choice is $P = \pm 1$ and $Q = 1$ since then we have $\alpha^3 = (\pm 1/2 + \sqrt{-3}/2)^3 = 1$. Moreover, it follows from $a^{p+1} \equiv \alpha\overline{\alpha} \equiv Q \pmod{p}$ that $\mathrm{ord}_p\alpha$ divides $2(p+1)$ if $Q \equiv \pm 1 \pmod{p}$. Since the maximal order of an element in $\mathcal{O}_{\sqrt{\Delta}}/(p)$ is $p^2 - 1$ we see that $Q \equiv \pm 1 \pmod{p}$ should generally be avoided. Table 3.1 shows the list of composite numbers $n < 80$ and the parameters $(P,Q)$ with $\left(\frac{\Delta}{n}\right) = -1$ for which these numbers are QF pseudoprimes. It can be seen that $Q = \pm 1$ occurs frequently.

The following two algorithms for choosing $P$ and $Q$ have been proposed in [BW80]. These algorithms choose the bases according to the comments in this section.

**Method $A^*$**

Step 1: *Let $D$ be the first element of the sequence $5, -7, 9, -11, 13, \ldots$ satisfying $\left(\frac{D}{n}\right) = -1$.*

Step 2: *If $D = 5$ then set $P \leftarrow 5$ and $Q \leftarrow 5$ otherwise set $P \leftarrow 1$ and $Q \leftarrow (1 - D)/4$.*

**Method $B^*$**

Step 1: *Let $D$ be the first element of the sequence $5, 9, 13, 17, 21, \ldots$ satisfying $\left(\frac{D}{n}\right) = -1$.*

Step 2: *Let $P$ be the smallest odd integer that is greater than $\sqrt{D+5}$ and set $Q \leftarrow (P^2 - D)/4$.*

It can be observed that the methods $A$ and $B$ in [BW80] eventually give $Q = \pm 1$. Therefore the methods $A^*$ and $B^*$ are probably stronger since these methods avoid $Q = \pm 1$.

## 3.5 Finding counterexamples to pseudoprimality tests

In this section we describe a method for finding strong pseudoprimes for a given set of small bases.

A composite number can be a strong pseudoprime for at most $1/4$ of all bases $b < n$ that are coprime to $n$. If a composite number $n$ is indeed a strong pseudoprime for $1/4$ of all bases then either $n$ is of the form $n = (k+1)\cdot(2k+1)$ or $n$ is a Carmichael number with three factors that are congruent 3 modulo 4. The first form was used by Arnault [Arn91] to generate a 46 digit composite strong pseudoprime for the prime bases 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 and 31. In [Arn95] he uses an improved method to construct a 337 digit composite strong pseudoprime for all prime bases below 200. $n = (k+1)(2k+1)$ implies that $8n+1$ is a perfect square. An implementation of a probabilistic primality test can check this condition very efficiently. But no efficient test for numbers of the second form is known.

The method presented here will generate strong pseudoprimes of the second form for large sets of prime bases. The generated numbers $n$ have the additional property that if $n$ is a strong pseudoprime for the bases $b_1$ and $b_2$ then it is a strong pseudoprime for the base $b_1 b_2$. This follows immediately from a theorem of Malm [Mal77]. He proved that this property holds for all $n$ congruent 3 modulo 4. Moreover, Davenport proposed in [Dav92] to perform certain tests in addition to the Miller-Rabin test. Observing that for any prime $p$ one half of the bases satisfy $b^{((p-1)/2)} \equiv -1 \pmod{p}$, he suggests to search for a base $b$ satisfying $b^{(n-1)/2} \equiv -1 \pmod{n}$ before $n$ is called "probably prime". Carmichael numbers with three factors that are congruent 3 modulo 4 satisfy the same condition and can therefore not be distinguished from primes by this additional test. Another suggestion by Davenport is to check whether more than two roots of -1 $\pmod{n}$ occur during the calculations of the Miller-Rabin test. If this happens then $n$ is composite. This test cannot be used when $n \equiv 3 \pmod{4}$ because -1 has no square roots in this case.

Arnault [Arn93] used a method similar to ours for finding strong pseudoprimes. His method can also be used to find strong pseudoprimes with more than 3 prime factors or strong Lucas pseudoprimes. However, some of his conditions for pseudo primality are too restrictive. For example no number of the form $(x+1)(5x+1)(21x+1)$ satisfies Lemma 3.1 of [Arn93], but it is this specific form of numbers which provided some of the best (i.e. smallest) counter-examples presented here. Hardly any number we found satisfies Arnault's restrictions. As a consequence our counter-examples for a given set of bases are much smaller than Arnault's.

**Lemma 3.16** *The following three conditions are equivalent for any primes $p_1, p_2, p_3$.*

*(i)* $p_1 p_2 p_3$ *is a Carmichael number*

*(ii)* $\operatorname{lcm}(p_1-1, p_2-1, p_3-1)$ *divides* $p_1p_2p_3-1$

*(iii) There exist integers* $a_1, a_2, a_3$ *and* $k$ *with* $a_1, a_2$ *and* $a_3$ *pairwise coprime,* $p_i = a_ik+1$ *for* $i = 1,2,3$ *and*

$$k(a_1a_2 + a_1a_3 + a_2a_3) + a_1 + a_2 + a_3 \equiv 0 \pmod{a_1a_2a_3} \tag{3.16}$$

*Proof:* The equivalence of $(i)$ and $(ii)$ has been shown by Carmichael [Car12]. To prove that $(iii)$ implies $(ii)$ we note that $\operatorname{lcm}(p_1-1, p_2-1, p_3-1) = \operatorname{lcm}(a_1k, a_2k, a_3k) = a_1a_2a_3k$ since $a_1, a_2$ and $a_3$ are pairwise coprime. Therefore we have

$$\begin{aligned} p_1p_2p_3 - 1 &\equiv (a_1k+1)(a_2k+1)(a_3k+1) - 1 \\ &\equiv k(k(a_1a_2 + a_1a_3 + a_2a_3) + a_1 + a_2 + a_3) \\ &\equiv 0 \pmod{a_1a_2a_3k}. \end{aligned}$$

The last congruence followed from Equation 3.16.

To prove that $(ii)$ implies $(iii)$ let $k = \gcd(p_1-1, p_2-1, p_3-1)$ and $a_i = (p_i-1)/k$. We will show that any integer $t$ dividing two of the three numbers $p_1-1, p_2-1$ and $p_3-1$ divides the third one too. Assume that $p_1 \equiv 1 \pmod t$ and $p_2 \equiv 1 \pmod t$. Then it follows that $n \equiv p_3 \pmod t$. From $t|(p_1-1)$ it follows that $t|(n-1)$ and hence that $1 \equiv n \equiv p_3 \pmod t$. Hence $a_1, a_2$ and $a_3$ are pairwise coprime. Thus we have $\operatorname{lcm}(p_1-1, p_2-1, p_3-1) = a_1a_2a_3k$ and (ii) implies

$$\begin{aligned} (p_1-1)(p_2-1)(p_3-1) - 1 \equiv\ & k(k(a_1a_2 + a_1a_3 + a_2a_3) \\ & + a_1 + a_2 + a_3) \pmod{a_1a_2a_3k} \end{aligned}$$

and this implies equation (3.16). $\square$

In the sequel we shall derive conditions on $k$ such that $n = p_1p_2p_3 = (a_1k+1)(a_2k+1)(a_3k+1)$ is a Carmichael number that is a strong pseudoprime for a given set of bases. We will discuss this only for Carmichael numbers whose prime factors are congruent 3 modulo 4.

Note that $p_i \equiv 3 \pmod 4$ for $i = 1,2,3$ implies $a_ik \equiv 2 \pmod 4$ and since the $a_i$'s are pairwise coprime, at most one of them can be even. Hence

$$k \equiv 2 \pmod 4 \tag{3.17}$$

and therefore $a_1, a_2$ and $a_3$ are all odd.

**Theorem 3.17** *Let* $n = p_1p_2p_3$ *be a Carmichael number with* $p_1 \equiv p_2 \equiv p_3 \equiv 3 \pmod 4$*. Then* $n$ *is a strong pseudoprime for the base* $b$ *if and only if*

$$\left(\frac{b}{p_1}\right) = \left(\frac{b}{p_2}\right) = \left(\frac{b}{p_3}\right) \neq 0 \tag{3.18}$$

*Proof:* From $n \equiv 3 \pmod 4$ it follows that $n$ is a strong pseudoprime for the base $b$ if and only if $b^{(n-1)/2} \equiv \pm 1 \pmod n$. This condition is equivalent to $b^{(n-1)/2} \equiv 1 \pmod{p_i}$ for all $i$ or $b^{(n-1)/2} \equiv -1 \pmod{p_i}$ for all $i$. The numbers $t_i := (n-1)/(p_i - 1)$ are odd because of $p_i \equiv 3 \pmod 4$ and $n \equiv 3 \pmod 4$. Thus $b^{(n-1)/2} \equiv b^{((p_i-1)/2)t_i} = \left(\frac{b}{p_i}\right)^{t_i} = \left(\frac{b}{p_i}\right) \pmod{p_i}$. □

**Theorem 3.18** *Let* $n = p_1p_2p_3 = (a_1k+1)(a_2k+1)(a_3k+1)$ *be a Carmichael number with* $p_i \equiv 3 \pmod 4$ *where the* $a_i$*'s are pairwise coprime. Then* $n$ *is a strong pseudoprime for the base 2 if and only if*

$$a_1 \equiv a_2 \equiv a_3 \pmod 4 \tag{3.19}$$

*Proof:* Let $k'$ be defined by $k = 4k'+2$. It follows that $\left(\frac{2}{p_i}\right) = (-1)^{(p_i^2-1)/8} = (-1)^{(8a_ik'+4a_i^2+4a_i)/8} = (-1)^{k'}(-1)^{(a_i^2+a_i)/2}$. Hence $\left(\frac{2}{p_i}\right) = (-1)^{k'+1}$ if $a \equiv 1 \pmod 4$ and $\left(\frac{2}{p_i}\right) = (-1)^{k'}$ if $a \equiv 3 \pmod 4$. The proof is completed by application of Theorem 3.17. □

**Theorem 3.19** *Let* $n = p_1p_2p_3 = (a_1k+1)(a_2k+1)(a_3k+1)$ *be a Carmichael number,* $p_1 \equiv p_2 \equiv p_3 \equiv 3 \pmod 4$*, and the* $a_i$*'s pairwise coprime. Let further* $b$ *be an odd integer,* $k \equiv m \pmod b$*. Then* $n$ *is a strong pseudoprime for the base* $b$ *if and only if*

$$\left(\frac{a_1m+1}{b}\right) = \left(\frac{a_2m+1}{b}\right) = \left(\frac{a_3m+1}{b}\right) \neq 0 \tag{3.20}$$

*Proof:* Let $k = bk' + m$. It follows by the quadratic reciprocity law that $\left(\frac{b}{p_i}\right) = \left(\frac{b}{a_ik+1}\right) = \left(\frac{a_ik+1}{b}\right)(-1)^{a_ik(b-1)/4} = \left(\frac{a_i(bk'+m)+1}{b}\right)(-1)^{k(b-1)/4} = \left(\frac{a_im+1}{b}\right)(-1)^{k(b-1)/4}$. Thus equation (3.18) is satisfied if and only if (3.20) is satisfied, since $(-1)^{k(b-1)/4}$ does not depend on $i$. □

**An example.** Let $a_1 = 1$, $a_2 = 5$, $a_3 = 9$, which are the smallest $a_i$ satisfying (3.19). Thus the numbers searched for have the form $n = (k+1)(5k+1)(9k+1)$. From equation (3.16) we get

$$k \equiv 15 \pmod{45} \tag{3.21}$$

(3.21) implies $k \equiv 0 \pmod 3$ and therefore $\left(\frac{a_ik+1}{3}\right) = 1$ for all $a_i$. Thus it follows from Theorem 3.19 that any Carmichael number of the form $n = (k+1)(5k+1)(9k+1)$ satisfying (3.21) is a strong pseudoprime for the base 3. Using a similar argumentation it can be shown that any such Carmichael number is a strong pseudoprime for the base 5. The conditions of Theorem 3.19 for $b = 7$ and $b = 11$ are satisfied if

$$\begin{aligned} k &\equiv 0 \pmod 7 \\ k &\equiv 0 \text{ or } 1 \pmod{11} \end{aligned}$$

We can combine these equations and (3.17) by using the Chinese Remainder Theorem. $(k+1)(5k+1)(9k+1)$ is a Carmichael number (with $p_i \equiv 3 \pmod 4$) which is a strong pseudoprime for all bases in the set $2, 3, 5, 7, 11$ if and only if the three factors are prime and $k \equiv 2310 \pmod{13860}$ or $k \equiv 4830 \pmod{13860}$.

By searching only through the first 50 values of $13860k' + 2310$ and $13860k' + 4830$ the following 5 solutions are found. The first one is a strong pseudoprime for the bases 13 and 17 too.

| $k$ | $(k+1)(5k+1)(9k+1)$ | $=$ | $n$ |
|---|---|---|---|
| 46410 | $46411 \cdot 232051 \cdot 417691$ | $=$ | 4498414682539051 |
| 390390 | $390391 \cdot 1951951 \cdot 3513511$ | $=$ | 2677380067596984751 |
| 404250 | $404251 \cdot 2021251 \cdot 3638251$ | $=$ | 2972788471124876251 |
| 503790 | $503791 \cdot 2518951 \cdot 4534111$ | $=$ | 5753899501012293751 |
| 614670 | $614671 \cdot 3073351 \cdot 5532031$ | $=$ | 10450558282397880151 |

Since (3.20) is satisfied for about 1/4 of all residue classes the number of possible residue classes grows exponentionally with the size of the set of bases. We have therefore chosen the following method for the construction of our largest examples. The set of bases is divided into four sets $B_1, B_2, B_3, B_4$. Let $N_i = \prod_{b \in B_i} b$ and let $R_i$ be the set of residues modulo $N_i$ satisfying (3.20) for all bases in $B_i$. $B_1$ and $B_2$ are chosen such that $R_1$ and $R_2$ have a size of 10,000 to 100,000. $N_3$ is approximately 1000 and $B_4$ contains the remaining bases. We wish to compute all $0 \le x < N_1 N_2$ that satisfy (3.20) for all bases in $B_1 \cup \cdots \cup B_4$. First two sets $S_1, S_2$ are computed by

$$S_1 := \{0 \le s < N_1 N_2 : \exists r \in R_1 : r \equiv s \pmod{N_1} \text{ and } s \equiv 0 \pmod{N_2}\}$$

$$S_2 := \{0 \le s < N_1 N_2 : \exists r \in R_2 : r \equiv s \pmod{N_2} \text{ and } s \equiv 0 \pmod{N_1}\}$$

All $0 \le x < N_1 N_2$ satisfying (3.20) for all bases in $B_1 \cup B_2$ can now be represented as $s_1 + s_2$ or $s_1 + s_2 - N_1 N_2$ for some $s_1 \in S_1$ and $s_2 \in S_2$. The elements of $S_2$ are now divided into the residue classes modulo $N_3$, i.e. the sets $C_0, \ldots, C_{N_3 - 1}$ are computed by $C_i = \{s \in S_2 : s \equiv i \pmod{N_3}\}$. Now for all $s_1$ in $S_1$ and $r_3$ in $R_3$ and $s_2 \in C_{r_3 - s_1 \bmod N_3}$ we check whether $x = s_1 + s_2$ satisfies (3.20) for all $p \in B_4$. Additionally we check for all $s_1$ in $S_1$ and $r_3$ in $R_3$ and $s_2 \in C_{r_3 - s_1 + N_1 N_2 \bmod N_3}$ whether $x = s_1 + s_2 - N_1 N_2$ satisfies $x > 0$ and (3.20) for all $p \in B_4$.

The smallest known strong pseudoprimes for all bases up to a given bound are listed in table 3.2. The first 4 strong pseudoprimes are contained in [PSW80]. Jaeschke asserts that the first 7 numbers are the smallest strong pseudoprimes for the given set of bases [Jae93]. The rest of the table was computed on a Sun-4 using the MAPLE algebra system taking a total of 200 hours of CPU time. The number that is a strong pseudoprime for all bases $\le 100$ was found using $a_1 = 1, a_2 = 5$ and $a_3 = 21$.

| $t$ | $n$ |
|---:|---|
| 2 | $2047 = 23 \cdot 89$ |
| 4 | $1373653 = 829 \cdot 1657$ |
| 6 | $25326001 = 2251 \cdot 11251$ |
| 10 | $3215031751 = 151 \cdot 751 \cdot 28351$ |
| 12 | $2152302898747 = 6763 \cdot 10627 \cdot 29947$ |
| 16 | $3474749660383 = 1303 \cdot 16927 \cdot 157543$ |
| 22 | $341550071728321 = 10670053 \cdot 3201057$ |
| 28 | $254699850156491854531 = 3561091 \cdot 100422739 \cdot 712219$ |
| 30 | $103705561641683704657 51 = 57230671 \cdot 1395871 \cdot 129815911$ |
| 36 | $4273439182293937563735 67 = 180503383 \cdot 222974767 \cdot 10617847$ |
| 40 | $1897597106867196176298868831$ <br> $= 2113672831 \cdot 12317610631 \cdot 72885271$ |
| 42 | $17554320317784476249592 6506791$ <br> $= 10076063791 \cdot 775081831 \cdot 22477373071$ |
| 46 | $1483664257341326129862007876051$ <br> $= 691604371 \cdot 44954284051 \cdot 47720701531$ |
| 52 | $168790877523676911809192454171451$ <br> $= 135547039651 \cdot 4674035851 \cdot 266420043451$ |
| 58 | $1088781536295680823159869241893780851$ <br> $= 872366473291 \cdot 174473294659 \cdot 7153405080979$ |
| 60 | $6737122462074619354527015408434000671$ <br> $= 7557029774911 \cdot 229000902271 \cdot 3893015338591$ |
| 70 | $167362055430542431223639618151191172751$ <br> $= 15336032025751 \cdot 17789797149871 \cdot 613441281031$ |
| 78 | $2909730361259578092634215960333442009 7591$ <br> $= 472363451220431 \cdot 2176790097791 \cdot 28298271271271$ |
| 82 | $1943507923865720818249653583964807114160978140504971$ <br> $= 132262179961786891 \cdot 264524359923573 79$ <br> $\cdot 555501155839504939$ |
| 88 | $6261592193697586271292194012879071219227335876980762 0171$ <br> $= 1767601880146745 5339 \cdot 841715181022259779$ <br> $\cdot 420857590511129 8891$ |
| 100 | $6852866339504691224422360590273835671975108278438668 1071$ <br> $= 182157454525892596 39 \cdot 433708225061649039 1$ <br> $\cdot 867416450123298079$ |

Table 3.2: Strong pseudoprimes for all bases $\leq t$

### 3.5.1 The size of strong pseudoprimes for all bases up to $t$.

Assuming the extended Riemann hypothesis Bach [Bac85] showed that no composite number passes the Miller-Rabin test for all bases $b < 2(\log N)^2$. Most implementations of the Miller-Rabin test use fewer bases. It has been shown [AGP94a] that there exist infinitely many composite numbers that are strong pseudoprime for any fixed set of bases. Moreover, the authors give heuristic reasons (based on the prime k-tuple conjecture) that there exist infinitely many composite numbers $n$ that are strong pseudoprimes for the first $f(n)$ prime numbers if $f(x) = o(\log(x))$. On the other hand it might be possible that there exists a constant $c$ such that no strong pseudoprime for the smallest $\lfloor c \cdot \log n \rfloor$ primes does exist. Davenport conjectured that for the choice $c = 1/\log 100$ no counter-example can be constructed. Our 41-digit number

$$29097303612595780926342159603334420097591$$

is a strong pseudoprime for the smallest 21 primes and is a counter-example to Davenport's conjecture.

## 3.6 Finding all pseudoprimes up to a given limit

In the previous section we described algorithms that could be used to find examples of pseudoprimes. However, these algorithms cannot be used to find a complete list of all pseudoprimes up to a given limit. Indeed, finding complete lists is the purpose of this section. Such lists can be used to check the quality of theoretical bounds. Alternatively, the goal of the section can be seen as finding a pseudoprimality test that is correct for all number up to this limit. For example minimal sets of bases for the Miller-Rabin test for certain limits will be presented.

### 3.6.1 Conditions on pseudoprimes

**Theorem 3.20** *Let $m \geq 1$ be an integer and $p$ be prime. If $n = mp^2$ is a Fermat pseudoprime for the base $b$ then the following two conditions must be satisfied.*

*(i)* $b^{p-1} \equiv 1 \pmod{p^2}$

*(ii)* $b^{m-1} \equiv 1 \pmod{p^2}$

*Proof:* It follows from $b^{n-1} \equiv 1 \pmod{n}$ that $b^{n-1} \equiv 1 \pmod{p^2}$. Hence $\mathrm{ord}_{p^2}(b)$ divides $n-1$. Since $\mathrm{ord}_{p^2}(b)$ also divides the group order $p(p-1)$ of $\mathbb{Z}/(p^2)^*$ it follows that $\mathrm{ord}_{p^2}(b)$ divides $\gcd(n-1, p(p-1))$. Since $p$ is relatively prime to $n-1$ it follows that $\gcd(n-1, p(p-1))$ divides $p-1$ and hence we have $b^{p-1} \equiv 1 \pmod{p^2}$. This proves (i).

Note that (ii) is trivialy satisfied when $m = 1$. Thus we may assume now that $m > 1$. Then $\mathrm{ord}_{p^2}(b)$ divides both $mp^2 - 1$ and $p-1$ and therefore $\mathrm{ord}_{p^2}(b)$ divides $(mp^2-1) - m(p+1)(p-1) = m-1$. Hence $b^{m-1} \equiv 1 \pmod{p^2}$. □

A prime $p$ that satisfies $b^{p-1} \equiv 1 \pmod{p^2}$ is called a *Wieferich prime*. Such primes have been studied by Lehmer [Leh81] who showed that there are no Wieferich primes for the bases 2 and 3 less than $T_1 = 6 \cdot 10^9$. Moreover, we have verified by an explicit calculation that no divisor of $gcd(2^{m-1}-1, 3^{m-1}-1)$ for $m \le T_2 = 100,000$ is a Wieferich prime for the bases 2 and 3. Thus we have

**Corollary 3.21** *All Fermat pseudoprimes $\le T_1^2 = 3.6 \cdot 10^{18}$ for the bases 2 and 3 are squarefree. Moreover, all Fermat pseudoprimes $\le T_1^2 T_2 = 3.6 \cdot 10^{24}$ are either squarefree or perfect squares.*

Theorem 3.20 shows that Fermat pseudoprimes containing a square up to a given limit $L$ can be found efficiently by searching for Wieferich primes up to $L^{1/2}$. We will therefore concentrate in the sequel on algorithms for finding squarefree Fermat pseudoprimes.

### 3.6.2 Pseudoprimes $n$ containing a given factor $m$

In this section we fix an odd squarefree integer $m$ and consider necessary conditions for an odd squarefree multiple $n = mw$ to be a pseudoprime or a strong pseudoprime for given base $b$ or for a given set of bases $b_1, b_2, \ldots, b_r$. The general case and the special case where $n$ is a prime multiple of $m$ are treated in subsection 3.6.2 and 3.6.3.

#### Without further conditions on $n$

Let $n = wm$ be an odd squarefree integer. By definition $n$ is a pseudoprime for the base $b$ if and only if $b^{wm-1} \equiv 1 \pmod{n}$, which implies $b^{wm-1} \equiv 1 \pmod{m}$. Hence $wm-1$ is a multiple of $\mathrm{ord}_m(b)$, i.e.,

$$w \equiv m^{-1} \pmod{\mathrm{ord}_m(b)} \tag{3.22}$$

Analogously we must have $w \equiv m^{-1} \pmod{\mathrm{ord}_w(b)}$.

For $n = wm$ to be a *strong* pseudoprime for the base $b$, $n$ and hence $m$ must be $b$-admissible, i.e., every prime factor $q$ of $n$ (and hence of $m$) must satisfy $\nu_2(\mathrm{ord}_q(b)) = c$ for $c = \nu_2(\mathrm{ord}_m(b))$. Let $w = \prod_{i=1}^{t} p_i$. Note that $\nu_2(\mathrm{ord}_{p_i}(b)) = c$ implies that $2^c$ divides $\mathrm{ord}_{p_i}(b)$ and therefore also $p_i - 1$. A consequence of Theorem 2.10 (iii) is that $\nu_2(\mathrm{ord}_{p_i}(b)) = c$ implies either

$$\left(\frac{b}{p_i}\right) = -1 \text{ and } \nu_2(p_i - 1) = c$$

or

$$\left(\frac{b}{p_i}\right) = 1 \text{ and } \nu_2(p_i - 1) > c.$$

Hence we have

$$p_i \equiv \begin{cases} 2^c + 1 \pmod{2^{c+1}} & \text{if} \quad \left(\frac{b}{p_i}\right) = -1 \\ 1 \pmod{2^{c+1}} & \text{if} \quad \left(\frac{b}{p_i}\right) = 1 \end{cases}$$

Note that $c = 0$ implies $p_i \equiv 1 \pmod{2^{c+1}}$ since we are considering only odd primes. Moreover, for $c > 0$ we have $(2^c + 1)^r \equiv 1 \pmod{2^{c+1}}$ if and only if $r$ is even. Thus it follows that $w = \prod_{i=1}^{t} p_i \equiv 1 \pmod{2^{c+1}}$ if the number of primes $p_i$ with $\left(\frac{b}{p_i}\right) = -1$ is even and $w \equiv 2^c + 1 \pmod{2^{c+1}}$ otherwise. Altogether we find that $w$ must satisfy (3.23) if $mw$ is a strong pseudoprime for base $b$.

$$w \equiv \begin{cases} 2^c + 1 \pmod{2^{c+1}} & \text{if} \quad \left(\frac{b}{w}\right) = -1 \\ 1 \pmod{2^{c+1}} & \text{if} \quad \left(\frac{b}{w}\right) = 1 \end{cases} \tag{3.23}$$

Note that this condition is only necessary but not sufficient.

Equations (3.23) and (2.12) together provide a restriction for the residue classes modulo $\mathrm{lcm}(4b, 2^{c+1})$ that $w$ can be a member of. These considerations can be combined if several bases $b_1, \ldots, b_r$ are considered giving Theorem 3.22.

**Theorem 3.22** *Let $b_1, \ldots, b_r$ be a given set of bases, let $n = mw$ be an odd squarefree integer where $m$ is $b_i$-admissible for $1 \leq i \leq r$, let $c_i = \nu_2(\mathrm{ord}_m(b_i))$ and let $Q$ be defined by*

$$Q = \mathrm{lcm}(4b_1, \ldots, 4b_r, 2^{c_1+1}, \ldots, 2^{c_r+1}).$$

*Then a necessary condition for $n$ to be a strong pseudoprime is that $w \equiv x \pmod{Q}$ where $x$ satisfies*

$$x \equiv \begin{cases} 2^{c_i} + 1 \pmod{2^{c_i+1}} & \textit{if} \quad \left(\frac{b_i}{x}\right) = -1 \\ 1 \pmod{2^{c_i+1}} & \textit{if} \quad \left(\frac{b_i}{x}\right) = 1 \end{cases}$$

*for all $1 \leq i \leq r$.*

*Example:* For $b = 2$ and $b = 3$ we get the following restrictions

| $b = 2$ | |
|---|---|
| $c$ | $w$ |
| 0 | $w \equiv 1, 7 \pmod{8}$ |
| 1 | $w \equiv 1, 3 \pmod{8}$ |
| 2 | $w \equiv 1, 5 \pmod{8}$ |
| $> 2$ | $w \equiv 1 \pmod{2^{c+1}}$ |

| $b = 3$ | |
|---|---|
| $c$ | $w$ |
| 0 | $w \equiv 1, 11 \pmod{12}$ |
| 1 | $w \equiv 1, 7 \pmod{12}$ |
| $\geq 2$ and even | $w \equiv 1, 2^c + 1 \pmod{3 \cdot 2^{c+1}}$ |
| $\geq 2$ and odd | $w \equiv 1, 5 \cdot 2^c + 1 \pmod{3 \cdot 2^{c+1}}$ |

Let $m = 117457 = 73 \cdot 1609$. Then it follows $\operatorname{lcm}(\operatorname{ord}_2(m), \operatorname{ord}_3(m)) = \operatorname{lcm}(603, 804) = 2412$). Hence for $mw$ to be a pseudoprime for the bases 2 and 3 we find that $w \equiv 1/m \equiv 1597 \pmod{2412}$ is a necessary condition. In order that $mw$ is a strong pseudoprime for the bases 2 and 3 we find two additional restrictions: $w \equiv 1$ or $7 \pmod 8$ and $w \equiv 1$ or $5 \pmod{24}$. The combination of these congruences by the Chinese Remainder Theorem results in the condition $w \equiv 4009 \pmod{4824}$.

### 3.6.3 $n$ is a prime multiple of $m$

We now investigate conditions on a prime $p$ for $n = mp$ to be a pseudoprime for a given base $b$. (Note that even though we are only interested in squarefree numbers $n$ the results in this section are valid for any odd $n$.) The pseudoprimality condition $b^{mp-1} \equiv 1 \pmod{mp}$ implies that

$$b^{mp-1} \equiv b^{m(p-1)+m-1} \equiv b^{m-1} \equiv 1 \pmod p, \tag{3.24}$$

i.e., that $p$ divides $b^{m-1} - 1$. Therefore, when $r$ bases $b_1, \ldots, b_r$ are considered, $p$ must divide

$$\gcd(b_1^{m-1} - 1, \ldots, b_r^{m-1} - 1).$$

This can be used to find all possible candidates $p$ by factoring reasonably large numbers, as long as $m$ is small enough to allow the computation of the greatest common divisors in feasible time. An improvement for reducing the size of the exponents in the gcd-terms will be discussed in Section 3.6.5. Assuming $b_1 < \cdots < b_r$, an efficient method for computing the gcd is to compute $T_1 := b_1^{m-1} - 1$ first, then $T_2 := b_2^{m-1} - 1 \bmod T_1$, $T_3 := b_3^{m-1} - 1 \bmod \gcd(T_1, T_2)$, etc, and computing the result by $gcd(T_r, T_{r-1})$.

If $n = pm$ is to be a *strong* pseudoprime for the base $b$, then the condition (3.24) can be sharpened by considering which algebraic factors of $b^{n-1} - 1$ that $p$ must divide.

Let $m = u2^v + 1$ with $u$ odd be $b_i$-admissible for $1 \leq i \leq r$ and let $h(b, m)$ for $b$ relatively prime to $m$ be defined by

$$h(b, m) := \begin{cases} b^u - 1 & \text{if } \nu_2(\operatorname{ord}_m(b)) = 0 \\ b^{u2^{c-1}} + 1 & \text{if } \nu_2(\operatorname{ord}_m(b)) = c > 0 \end{cases}$$

and otherwise by $h(b, m) := 1$. Thus we have

**Theorem 3.23** *A necessary condition for an integer $n$ of the form $n = mp$ for a given $m$ with $p$ prime to be a strong pseudoprime for the bases $b_1, \ldots, b_r$ is that $p$ divides* $\gcd(h(b_1, m), \ldots, h(b_r, m))$.

Note that according to the definition of $h(b, m)$, the condition on $p$ in Theorem 3.23 implies that $m$ is $b_i$-admissible for $1 \leq i \leq r$.
*Proof:* Let $m = u2^v + 1$ with $u$ odd. From $b_i^{mp-1} \equiv 1 \pmod{mp}$ it follows that $b_i^{mp-1} \equiv b_i^{m(p-1)+m-1} \equiv b_i^{m-1} \equiv b_i^{u2^v} \equiv 1 \pmod{p}$. Let $c_i = \nu_2(\mathrm{ord}_m(b_i))$. Since $c_i = \nu_2(\mathrm{ord}_p(b_i)) = \nu_2(\mathrm{ord}_m(b_i))$ is a necessary condition $p$ divides $b_i^{u2^{c_i}} - 1$. Moreover, if $c_i > 0$ it follows that $p$ does not divide $b_i^{u2^{c_i-1}} - 1$ and therefore $p$ must divide $b_i^{u2^{c_i-1}} + 1$. □
*Example:* Consider the prime $m = 15101$. We have $\nu_2(\mathrm{ord}_m(2)) = 2$ and $\nu_2(\mathrm{ord}_m(3)) = 2$. It follows that $h(2, m) = 2^{7550} + 1$ and $h(3, m) = 3^{7550} + 1$. Therefore we have $\gcd(h(2, m), h(3, m)) = 34545993300125$ which factors as $5^3 \cdot 101 \cdot 15101 \cdot 181201$. One can easily verify that $15101 \cdot 181201$ is the only prime multiple of $m$ that is a strong pseudoprime for the bases 2 and 3.

### 3.6.4 Outline of the algorithm

The algorithm outlined in this section finds all strong pseudoprimes for the bases $b_1, \ldots, b_r$ up to the limit $L$. Let $\pi_1(n)$ denote the largest prime factor of $n$, and let $K < \sqrt{L}$ be a given parameter. The algorithm proceeds in two steps with $K \approx L^{0.3}$. In the first and the second step it finds the strong pseudoprimes $\leq L$ with $\pi_1(n) < n/K$ and $\pi_1(n) \geq n/K$, respectively.

In the first step we apply the results of Section 3.6.2, where $m$ ranges over all integers in the interval $[K, \sqrt{L}]$ that are $b_i$-admissible for $1 \leq i \leq r$ and which have no proper divisor greater than $K$. The last condition can be put on $m$ because every composite $n < L$ with $\pi_1(n) < n/K$ has a divisor in the interval $[K, \sqrt{L}]$. Therefore, if some $m_1$ has a divisor $m_0 > K$ then every strong pseudoprime of the form $m_1 w_1$ is also found by the search for strong pseudoprimes of the form $m_0 w_0$.

Every $m \in [K, \sqrt{L}]$ with no proper divisor greater than $K$ that is $b_i$-admissible for $1 < i \leq r$ is either a prime or the product $m = p_1 \cdots p_t$ of primes $\leq K$ with $\nu_2(\mathrm{ord}_{p_j}(b_i)) = \nu_2(\mathrm{ord}_{p_k}(b_i))$ for $1 < i \leq r$ and $1 \leq j < k \leq t$. All integers $m$ of the latter form can be obtained by partitioning the set of primes $p \leq K$ into disjoint subsets for which the values $\nu_2(\mathrm{ord}_p(b_i))$ for $1 < i \leq r$ agree, and computing the integers in $[K, \sqrt{L}]$ consisting of distinct prime factors from only one of the subsets.

For each $m$ of one of the described forms the possible residue classes modulo

$$Q = \mathrm{lcm}(4b_1, \ldots, 4b_r, 2^{\nu_2(\mathrm{ord}_m(b_1))+1}, \ldots, 2^{\nu_2(\mathrm{ord}_m(b_r))+1})$$

for $w$ are found according to Theorem 3.22. Then one checks for all $w \leq L/m$ in the possible residue classes whether $mw$ is a strong pseudoprime for the bases $b_1, \ldots, b_r$. This procedure finds all strong pseudoprimes $\leq L$ with $\pi_1(n) < n/K$.

In the second step we apply the results of Section 3.6.3, where $m$ ranges over all integers $\leq K$ that are $b_i$-admissible for $1 \leq i \leq r$, i.e., that consist of distinct prime factors from only one of the subsets.

### 3.6.5 Further Improvements

A well-known theorem of Legendre (see for example [Rie85, p. 184]) states that if a prime $p$ divides $a^n - 1$, then either $p = kn + 1$ for some integer $k$ or $p$ divides an algebraic factor of $a^n - 1$. This theorem can be generalized as follows:

**Theorem 3.24** *Let $p$ and $q$ be primes where $p$ divides $a^{nq^\alpha} - 1$ and let $0 \leq \beta < \alpha$. Then either $p$ divides $a^{nq^\beta} - 1$ or $p = kq^{\beta+1} + 1$ for some integer $k$.*

*Proof:* If $\mathrm{ord}_p(a)$ divides $nq^\beta$ then $p$ divides $a^{nq^\beta} - 1$. Otherwise, if $\mathrm{ord}_p(a)$ does not divide $nq^\beta$, it follows that $q^{\beta+1}$ divides $\mathrm{ord}_p(a)$ since $\mathrm{ord}_p(a)$ divides $nq^\alpha$. □

This theorem can be used to improve the second step of the algorithm, in which one is given an integer $m$ and wants to compute the set $T_2$ of strong pseudoprimes of the form $n = mp$. If $m$ is close to $K$, a reduction of the exponents in the terms $h(b, m)$ can be achieved, which simplifies the *gcd* computation.

The pseudo-primality condition $b^{mp-1} \equiv 1 \pmod{mp}$ is equivalent to the two conditions $p|(b^{m-1} - 1)$ and $p \equiv 1/m \pmod{\mathrm{ord}_m(b)}$. It follows from Theorem 3.10 that $mp$ is a strong pseudoprime for the base $b$ if and only if, in addition to these equations we have $\nu_2(\mathrm{ord}_m(b)) = \nu_2(\mathrm{ord}_p(b)) := c$. If $m$ is of the form $m = lq^\beta + 1$ where $q$ is prime (and preferably large), we can apply Theorem 3.24 to replace the condition $p|(b^{m-1} - 1)$ by two alternative conditions that are computationally easier to check: $p|(b^{m-1} - 1)$ if and only if $p \equiv 1 \pmod q$ or $p|(b^l - 1)$. Similar arguments as used in Theorem 3.23 can now be used to simplify $p|(b^l - 1)$ further. Let $l = u2^v$ with $u$ odd. In order for $mp$ to be a strong pseudoprime for the base $b$, one of the three conditions (3.25), (3.26) or (3.27) must be satisfied.

$$p \equiv 1 \pmod q \tag{3.25}$$

$$\text{or} \qquad c = 0 \quad \text{and} \quad p|(b^u - 1) \tag{3.26}$$

$$\text{or} \qquad c > 0 \quad \text{and} \quad p|(b^{u2^{c-1}} + 1) \tag{3.27}$$

Another improvement was used to compute $\gcd(2^{m+a} - \delta, 3^{m+a} - \epsilon)$ for a fixed $m$ and different values of $a$ that are small compared to $m$. First by computing the continued fraction expansion of $(2/3)^m$ integers $x_1, y_1, z_1$ and $x_2, y_2, z_2$ are computed (i.e. $(2/3)^m \approx x_i/y_i$) such that the sizes of $y_i$ and $z_i$ are almost equal and

$$3^m x_i - 2^m y_i = z_i$$

for $i \in \{1, 2\}$. Then $\gcd(2^{m+a} - \delta, 3^{m+a} - \epsilon)$ divides $2^a x_i(3^{m+a} - \epsilon) - 3^a y_i(2^{m+a} - \delta) = 6^a z_i - 2^a x_i \epsilon + 3^a y_i \delta$ for $i \in \{1, 2\}$. Thus if

$$g_a(\epsilon, \delta) := \gcd(6^a z_1 - 2^a x_1 \epsilon + 3^a y_1 \delta, 6^a z_2 - 2^a x_2 \epsilon + 3^a y_2 \delta)$$

then the gcd can be computed as

$$\gcd(2^{m+a} - \delta, 3^{m+a} - \epsilon) = \gcd(2^{m+a} - \delta \bmod g_a(\epsilon, \delta), 3^{m+a} - \epsilon \bmod g_a(\epsilon, \delta)).$$

In the case $\delta = \epsilon = 1$ it can be shown that $g_a(\epsilon, \delta) = \gcd(2^{m+a} - 1, 3^{m+a} - 1)$ if $x_1, y_1, z_1, x_2, y_2, z_2$ correspond to two consecutive fractions of the continued fraction expansion of $(2/3)^m$.

### 3.6.6 Comparison to related work

The algorithm presented in this chapter is similar to two algorithms proposed by Pinch and Jaeschke. Pinch describes in [Pin93c] an algorithm that finds all Fermat-pseudoprimes to one base up to a given limit. The aim of the algorithm presented by Jaeschke [Jae93] is to find the smallest number that is a strong pseudoprime for a given set of bases. Even though these three algorithms have the same goal (computing pseudoprimes) they are optimized for different constraints on the bases.

It should be noted that searching for all pseudoprimes to one base up to some limit is much harder than searching for pseudoprimes to a set of at least two bases. The crucial point is that the algorithm used by R. Pinch to find pseudoprimes to one base $b$ needs the factorization of numbers of the form $b^n - 1$ for $n$ as large as possible. If a set $\{b_1, b_2, \ldots\}$ of at least two bases is considered then it is only necessary to find the factorization of $\gcd(b_1^n - 1, b_2^n - 1)$ (see section 3.6.3). Since the gcd is much smaller than the arguments it is possible to compute these gcd's for much larger $n$. Because of these differences our algorithm can be used to search up to a larger limit.

The algorithm of Jaeschke searches for numbers that are even more restricted than ours. Since Jaeschke computes strong pseudoprimes to a set of bases that is rather large the second phase of the algorithm can be performed much faster. Consequently the algorithm described in this chapter tries to catch more pseudoprimes in the first phase by computing more gcd's. This was made possible by the improvements described in Section 3.6.5.

### 3.6.7 Numerical results

Table 3.3 is a compilation of the number of integers less than or equal to a given bound $L$ that are strong pseudoprimes for certain sets of small bases.

The number of strong pseudoprimes for the base 2 have been computed by Pinch [Pin93c]. Moreover, we have used the lists of all pseudoprimes up to $10^{13}$ and all Carmichael numbers up to $10^{16}$ [Pin93a],[Pin93b] computed by Pinch to verify partially our results.

We have tested the strong pseudoprimes found by our program against various versions of the Lucas test. No strong pseudoprime passes the Lucas test proposed in [BW80] regardless whether method $A, B, A^*$ or $B^*$ is used to find

| | set of bases | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| L | $B_2$ | $B_3$ | $B_5$ | $B_7$ | $B_{11}$ | $B_{13}$ | $B_{17}$ | $B_{19}$ |
| $10^6$ | 46 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $10^7$ | 162 | 7 | 0 | 0 | 0 | 0 | 0 | 0 |
| $10^8$ | 488 | 21 | 1 | 0 | 0 | 0 | 0 | 0 |
| $10^9$ | 1282 | 58 | 3 | 0 | 0 | 0 | 0 | 0 |
| $10^{10}$ | 3291 | 133 | 8 | 1 | 0 | 0 | 0 | 0 |
| $10^{11}$ | 8607 | 351 | 37 | 1 | 0 | 0 | 0 | 0 |
| $10^{12}$ | 22407 | 967 | 101 | 9 | 0 | 0 | 0 | 0 |
| $10^{13}$ | | 2696 | 240 | 24 | 2 | 1 | 0 | 0 |
| $10^{14}$ | | 7117 | 627 | 86 | 17 | 3 | 0 | 0 |
| $10^{15}$ | | 19322 | 1730 | 227 | 38 | 6 | 1 | 1 |
| $10^{16}$ | | 52593 | 4603 | 606 | 107 | 11 | 2 | 1 |

Table 3.3: The number of strong pseudoprimes up to $L$ for the set of bases $B_n$ of all primes up to $n$

the parameters $P$ and $Q$. As a consequence any of these tests combined with a Miller-Rabin test for the bases 2 and 3 gives a correct primality test up to $10^{16}$.

The function `PrimeQ` of Mathematica 2.3 uses a Miller-Rabin test with the bases $\{2,3\}$ and a version of the test shown in Section 3.2.5. It has been verified that none of the strong pseudoprimes found passes the modified test in Mathematica. It follows that the new test is correct at least up to $10^{16}$.

The computation of the pseudoprimes up to $10^{16}$ used approximately 60 hours CPU time on an Alpha-Workstation. Based on test runs up to $10^{14}$ we estimated that the modifications in section 3.6.5 saved about 60-70% of the total CPU time.

The running time for the first step seems to be dominated by the time used for the computation of the gcd's. The necessary factorizations after the gcd computation never proved to be hard, since the gcd's are most likely the product of small primes. Therefore we conjecture that the running time for the first step is about $O(K^3)$ (assuming an $O(n)^2$ algorithm for the computation of gcd of length $n$).

The running time for the second step depends on the number of pseudo primality tests that have to be performed. This number is of the order

$$\sum_x \frac{L}{x\,\mathrm{lcm}(\mathrm{ord}_x(b_1), \mathrm{ord}_x(b_2), \ldots)}$$

where $x$ runs over all $b_i$-admissible numbers in the interval $[K, L^{1/2}]$ that have no divisor larger than $K$. We have not been able to prove a good approximation for this formula. A rough guess that is only supported by numerical data is that the running time for the second step is about $O(LK^{-2/3})$. A consequence of this would be that $K = L^{3/11}$ is nearly optimal and that the running time of the algorithm under these assumptions is $O(L^{9/11})$.

# Chapter 4

# Addition chains

## 4.1 Definitions and motivation

In this chapter we will study the problem of computing powers efficiently. We will assume that the cost for one multiplication is close to constant, so that the time to compute a power depends essentially on the number of multiplications that have to be performed. This assumption is reasonable in many cases (e.g. when the powers are computed in a finite field or in $\mathbb{Z}/(m)$). One possible way is to compute $x^n$ by the *binary method*. Equation (4.1) is a recursive description of this method.

$$x^n = \begin{cases} 1 & \text{if } n = 0, \\ \left(x^{n/2}\right)^2 & \text{if } n > 0 \text{ is even }, \\ x^{n-1}x & \text{if } n > 0 \text{ is odd.} \end{cases} \tag{4.1}$$

This method is reasonably fast. For example only 7 multiplication are needed to compute $x^{23}$. The powers $x^2, x^4, x^5, x^{10}, x^{11}, x^{22}$ and $x^{23}$ are computed as intermediate results where each of these powers is computed from previous ones using one multiplication. However, the optimal way to compute $x^{23}$ requires only 6 multiplications since $x^{23}$ can be computed by

$$\begin{aligned} x^2 &= x \cdot x \\ x^3 &= x^2 \cdot x \\ x^5 &= x^3 \cdot x^2 \\ x^{10} &= x^5 \cdot x^5 \\ x^{20} &= x^{10} \cdot x^{10} \\ x^{23} &= x^{20} \cdot x^3. \end{aligned}$$

The goal of this chapter is to find the smallest number of multiplication to compute $x^n$ for a given $n$.

Let us look at the problem in an abstract way. Since $x^{i+j}$ can be computed from $x^i$ and $x^j$ with one multiplication we look for a chain of exponents such that each exponent is the sum of previous exponents. This justifies the following definition.

**Definition 4.1** *An* addition chain *for a positive integer $n$ is an ascending list of integers $a_0, \ldots, a_r$ such that $a_0 = 1$, $a_r = n$ and such that for all $1 \leq k \leq r$ there exists $i, j < k$ with $a_k = a_i + a_j$ The length of this chain is $r$. The minimal length of an addition chain for $n$ is denoted by $\ell(n)$. An addition chain for a set $S$ of positive integers is an addition chain containing every element of $S$. $\ell(S)$ denotes the length of a shortest addition chain for $S$.*

Note that a chain of length $r$ has $r + 1$ elements. The reason for this definition of length is that an addition chain of length $r$ corresponds to a way of computing $x^n$ with $r$ multiplications. Addition chains for sets will be used later. A short addition chain for a set $\{s_1, s_2, \ldots\}$ can be interpreted as a description to compute the set of powers $\{x^{s_1}, x^{s_2}, \ldots\}$ efficiently.
*Example:* The methods for computing $x^{23}$ shown earlier in this section correspond to the addition chains $1, 2, 4, 5, 10, 11, 22, 23$ and $1, 2, 3, 5, 10, 20, 23$, which have lengths 7 and 6, respectively.

Addition chains have been studied extensively. A good overview is given in [Knu81, p.441–466]. Downey, Leong and Sethi show that the problem of finding an optimal addition chain for a set $S$ is NP-complete [DLS81]. We are not aware of a proof that the problem of computing $\ell(n)$ given $n$ is NP-complete. However, it seems hard to compute optimal addition chains. Different heuristics for finding short, but not necessarily optimal addition chains for $n$ large have been proposed. Bos and Coster [BC90] use a window method. In addition they give a heuristic method for finding short chains for sets of integers. Yacobi [Yac90] uses a method similar to the Lempel-Ziv data compression algorithm. His method can also be seen as an extension of Bos and Coster's method by allowing windows of different size. A method for computing addition chains under memory restrictions is described in [SD93]. A method for describing the integers that have small addition chains has been described by Flammenkamp [Fla91].

The goal of this chapter is to describe a new algorithm for computing optimal addition chains efficiently. A valuable tool that can be used in many proofs is the graphical representation of addition chains. This tool has been introduced by Knuth [Knu81]. In Section 4.3 we describe a slightly different form of graphical representations, which has the advantage of being more general and more useful for addition chains for sets. Section 4.4 describes the outline of the new algorithm. The runtime of this algorithm strongly depends on the quality of heuristics for computing upper and lower bounds on the length of an addition chain. Methods for finding upper bounds are described in Section 4.5. Such upper bounds are generally computed by looking for a short addition chain using heuristics. It was our experience that known methods for finding short addition chains were sufficient for computing good upper bounds. Section 4.6 describes some methods for finding lower bounds. The bounds that can be derived from known theorems have not been good enough to create an efficient algorithm. Therefore we describe some new methods to find good lower bounds, which are generally based on previously computed results. Section 4.7 describes still another source that can be used to speed up the algorithm. If it is possible to show that any addition chain searched in one branch of the search tree can be transformed into an other addition chain of the same length in an other branch of the search tree then it is not necessary to search the first branch of the tree at all.

## 4.2 Binary addition chains

The *binary addition chain* $B(n)$ for $n$ corresponds to the exponents of the binary method for computing powers. It can therefore be described by

$$B(n) = \begin{cases} 1 & \text{if } n = 1 \\ B(n/2), n & \text{if } n \text{ is even} \\ B(n-1), n & \text{otherwise.} \end{cases} \tag{4.2}$$

**Lemma 4.2** *The length of the binary addition chain $B(n)$ is*

$$\lfloor \log_2(n) \rfloor + w(n) - 1$$

*where $w(n)$ denotes the number of bits in $n$ equal to 1.*

The lemma can be shown by induction over $n$. It follows immediately that

$$\ell(n) < 2\lfloor \log_2(n) \rfloor. \tag{4.3}$$

## 4.3 Graphical representation

The following representation of addition chains has been introduced by Knuth [Knu81, p.460ff]. Many transformations of addition chains are easier to understand when this representation is used.

A graph $G$ will be described by a quadruple $G = (V, E, \alpha, \omega)$ where $V$ is the set of vertices, $E$ is the set of edges and $\alpha, \omega : E \to V$ are two functions denoting the initial vertex $\alpha(e)$ and the terminal vertex $\omega(e)$ of an edge $e$. This notation allows to describe directed graphs with parallel edges (i.e. edges with the same initial and terminal nodes). The *in-degree* of a vertex $v$ denotes the number of edges to $v$ (i.e. in-degree$(v) = \#\{e \in E : \omega(e) = v\}$). The *out-degree* of a vertex $v$ denotes the number of edges from $v$ (i.e. out-degree$(v) = \#\{e \in E : \alpha(e) = v\}$). A *source* is a vertex with in-degree 0. If a graph $G$ is acyclic then the number of distinct paths between two vertices is finite, i.e. the number of paths $\mathrm{p}(v,w)$ from a vertex $v$ to a vertex $w$ can recursively be defined as

$$\mathrm{p}(v,w) = \begin{cases} 1 & \text{if} \quad v = w \\ \displaystyle\sum_{e \in E:\omega(e)=w} \mathrm{p}(v,\alpha(e)) & \text{if} \quad v \neq w \end{cases} \tag{4.4}$$

A close connection between acyclic directed graphs $G$ with one source and addition chains is shown by Theorem 4.3 and 4.4.

**Theorem 4.3** *For every addition chain $a_0,\ldots,a_r$ there exists a directed acyclic graph $G=(V,E,\alpha,\omega)$ with $r+1$ vertices $V=\{v_0,\ldots,v_r\}$ and $2r$ edges, such that $\mathrm{p}(v_0,v_i)=a_i$ for $0 \le i \le r$.*

*Proof:* We will prove the theorem by giving an explicit construction of a such a graph.

Since $a_0,\ldots,a_r$ is an addition chain we can find indices $s_i$ and $t_i$ for all $1 \le i \le r$ such that $a_i = a_{s_i} + a_{t_i}$. The graph $G$ can then be constructed by introducing one vertex $v_i$ for every $a_i$. For each $1 \le i \le r$ two edges from $v_{s_i}$ to $v_i$ and from $v_{t_i}$ to $v_i$ are drawn. i.e. $G=(V,E,\alpha,\omega)$ where $V=\{v_0,\ldots,v_r\}$, $E=\{e_1,\ldots,e_r,\tilde{e}_1,\ldots,\tilde{e}_r\}$, $\alpha(e_i)=v_{s_i}$, $\alpha(\tilde{e}_i)=v_{t_i}$ and $\omega(e_i)=\omega(\tilde{e}_i)=v_i$.

The relation $\mathrm{p}(v_0,v_i)=a_i$ follows by induction over $i$ since we have $\mathrm{p}(v_0,v_0) = 1 = a_0$ and for $1 \le i \le r$ we have $\mathrm{p}(v_0,v_i)=\mathrm{p}(v_0,v_{s_i})+\mathrm{p}(v_0,v_{t_i}) = a_{s_i}+a_{t_i} = a_i$. □

Two graphs representing the addition chains for $1,2,3,5,10,20,23$ and $1,2,3,5,10,13,23$ are shown in figure 4.1. Note that this does not define a one-to-one relation between acyclic directed graphs with one source and addition chains.

**Theorem 4.4** *For every directed acyclic graph $G=(V,E,\alpha,\omega)$ with source $w$ there exists an addition chain $a_0,\ldots,a_r$ of length at most $\#E-\#V+1$ for the set $\{\mathrm{p}(w,v) : v \in V\}$.*

*Proof:* First we will construct a set $S$ containing the set $\{\mathrm{p}(w,v) : v \in V\}$ such that for all $x \in S \setminus \{1\}$ there exist $y,z \in S$ with $x=y+z$. For all $v \in V$ we construct a set $A_v$ by

- $A_w = \{1\}$
- $A_v = \emptyset$ if the in-degree of $v$ is 1.
- For a vertex $v$ with in-degree $k \ge 2$ and successors[1] $v_1,\ldots,v_k$ we construct the set $A_v$ by

$$\begin{aligned} &\mathrm{p}(w,v_1)+\mathrm{p}(w,v_2), \\ &\mathrm{p}(w,v_1)+\mathrm{p}(w,v_2)+\mathrm{p}(w,v_3), \\ &\vdots \\ &\mathrm{p}(w,v_1)+\mathrm{p}(w,v_2)+\cdots+\mathrm{p}(w,v_r)=\mathrm{p}(w,v). \end{aligned}$$

[1] A vertex may be a multiple successor of $v$ if there are parallel edges from this vertex to $v$. Such a vertex would appear more than once in the list of successors.

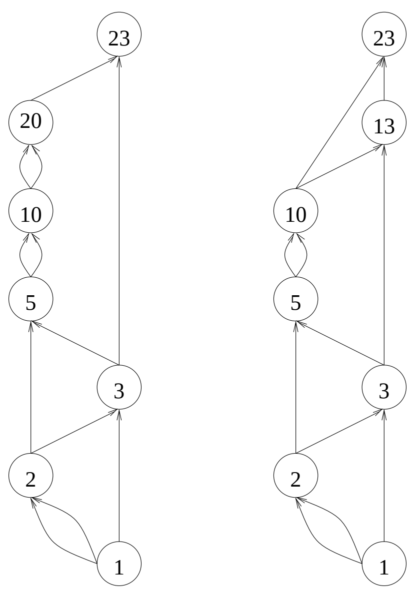


Figure 4.1: graphical representation of two addition chains for 23. Each label of a vertex $v$ corresponds to $\mathrm{p}(w, v)$ where $w$ is the source of the graph.

Let $S = \bigcup_{v \in V} A_v$. It follows that $\mathrm{p}(w,v)$ is an element of $S$ for all $v \in V$ because for vertices $v$ with in-degree 1 and successor $v'$ we have $\mathrm{p}(w,v) = \mathrm{p}(w,v')$ and because cycles are not possible in $G$. All elements of $S \setminus \{1\}$ are therefore the sum of two elements of $S$. Moreover, we have $\#A_v = \text{in-degree}(v) - 1$ for all $v \neq w$. Thus it follows that $\#S \leq \#E - \#(V \setminus \{w\}) + 1$, where we have equality when the sets $A_v$ are disjunct. Thus by sorting $S$ we get an addition chain of length $\#S - 1 \leq \#E - \#V + 1$. $\square$

**Definition 4.5** *A directed acyclic graph $G = (V, E, \alpha, \omega)$ with source $w$ is called* associated *to the addition chain $a_0, \ldots, a_r$ if $\#E - \#V + 1 = r$ and if for all $v \in V$ the value $\mathrm{p}(w,v)$ appears in the addition chain. A graph $G$ with source $w$ associated to an addition chain $a_0, \ldots, a_r$ for a set $S$ of positive integers is called* reduced*, if*

*(i) every vertex $v$ with out-degree smaller than 2 corresponds to an element $s \in S$ (i.e. $s = \mathrm{p}(w,v)$).*

*(ii) for all distinct $v, v' \in V$ we have $\mathrm{p}(w,v) \neq \mathrm{p}(w,v')$.*

*(This implies that every vertex except $w$ has an in-degree greater than or equal to 2.)*

*Moreover, we call two addition chains* equivalent *if they are associated to the same reduced graph.*

The addition chains $1, 2, 3, 5, 10, 20, 23$ and $1, 2, 3, 5, 10, 13, 23$ are equivalent since they have the same reduced graph, which is shown in Figure 4.2 on the left side. It is possible to find a reduced graph for every addition chain for a set $S$. Given a graph $G$ associated to the addition chain $A = a_0, \ldots, a_r$ a reduced graph can be constructed by iteratively deleting the vertices that do not satisfy Definition 4.5. Any vertex not corresponding to an element in $S$ having out-degree 0 can be deleted. If $v$ is a vertex not corresponding to an element in $S$ (i.e. $\mathrm{p}(w,v) \notin S$) having out-degree 1 then $v$ is deleted and every edge to $v$ is redirected to the successor of $v$. If $v$ is vertex with in-degree 1 then $v$ is deleted and the initial vertex of every edge from $v$ is changed to the predecessor of $v$. Note that this process deletes a least as many edges as vertices. Therefore we see that $\#E - \#V + 1$ is less than or equal to the length of the addition chain $A$.

One reason to introduce the concept of reduced graphs is that it allows to recognize certain similar chains. A program searching for a shortest addition chain has to consider only one chain for a given reduced graph.

**Definition 4.6** *Let $G = (V, E, \alpha, \omega)$ be a graph. The* dual *Graph $G^d$ of $G$ is the graph that is obtained by changing the direction of all edges in $G$, i.e.*

$$G^d = (V, E, \omega, \alpha)$$

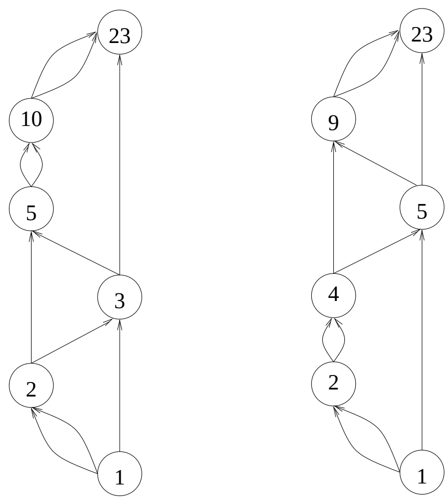


Figure 4.2: The reduced graph for the addition chains of the preceding figure and its dual graph. The values $\mathrm{p}(w, v)$ are elements of two different addition chains.

An important property of the dual graph is that the number of paths in the graph does not change. In particular let $G = (V, E, \alpha, \omega)$ be a directed acyclic graph and let $G^d = (V, E, \omega, \alpha)$ be its dual graph. Then for all $v, w \in V$ we have $\mathrm{p}(v, w) = k$ in $G$ if and only if $\mathrm{p}(w, v) = k$ in $G^d$.

## 4.4 Outline of the search algorithm

In this section we describe the outline of an algorithm for computing shortest addition chains. Since this algorithm takes advantage of previously computed information (e.g. the computation of $\ell(n)$ uses the values $\ell(m)$ with $m < n$) it is most efficient when it is used to compute $\ell(n)$ for all $n$ up to a given bound.

One method to find $\ell(S)$ is to compute it recursively from $\ell(S')$ where $\max(S') < \max(S)$. In particular let $S$ be a nonempty set not equal to $\{1\}$ and let $m$ the maximal element of $S$. Assume that $a_0, a_1 \ldots a_r$ is an optimal addition sequence for $S$. Then $a_r = m$ and $a_r = a_i + a_j$ for some $i, j < r$. It follows that the sequence $a_0, a_1 \ldots a_{r-1}$ is an optimal sequence for the set $S \cup \{a_i, a_j\} \setminus \{a_r\}$. Thus if $m \geq 2$ is the maximal element in $S$ then we have

$$\ell(S) = \min_{1 \leq x < m} \ell(S \cup \{x, m - x\} \setminus \{m\}) + 1 \tag{4.5}$$

A naive procedure that uses (4.5) to find the length of an optimal addition chain

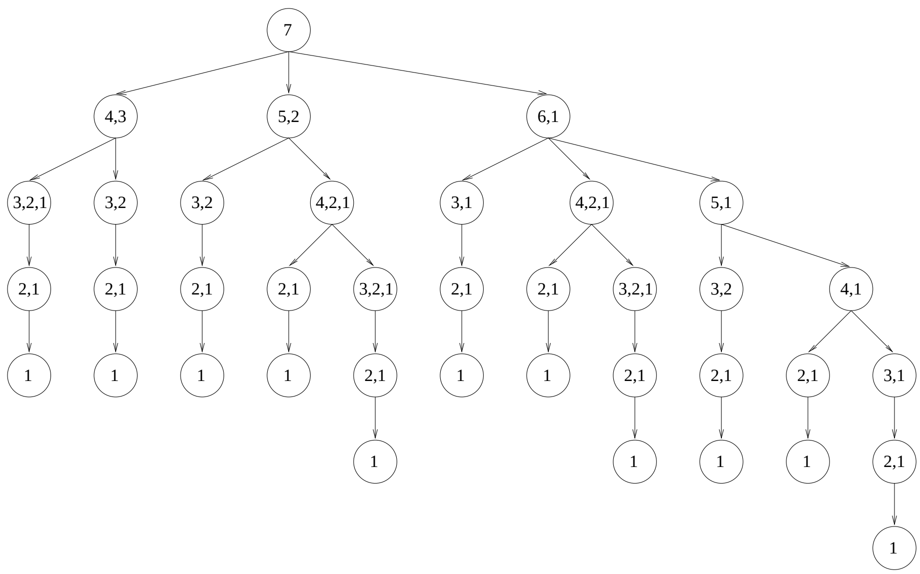


Figure 4.3: Search tree for $S = \{7\}$

for the set $\{7\}$ would produce the search tree shown in Figure 4.3. The vertices are labeled with the sets occurring in (4.5). An edge from $S_i$ to $S_j$ is drawn if $S_i$ can be computed from $S_j$ in one step. An addition chain for $\{7\}$ corresponds to a path from the root vertex to any leaf. The set of elements in this chain is the union of all sets along this path and the length of the chain is equal to the number of edges of the path.

A first improvement is to avoid that $\ell(S)$ is computed more than once or equivalently to avoid that a set occurs more than once in the search tree. This results in the search graph shown in Figure 4.4. Another improvement comes from the fact that 1 and 2 are contained in any addition chain for a set $S$ with $\max(S) \geq 2$. Therefore we have

$$\ell(S) = \ell(S \cup \{1, 2\}). \tag{4.6}$$

It can be seen from Figure 4.4 that the search graph for the set $\{7, 2, 1\}$ has fewer nodes than the search graph for $\{7\}$ since certain nodes are joined together (e.g. the nodes $\{5, 2\}$ and $\{5, 1\}$ correspond to the node $\{5, 2, 1\}$ in the search graph for $\{7, 2, 1\}$).

Since the size of this search graph grows very fast with $n$ a search algorithm is only efficient when the search graph can be pruned rigorously. Methods for efficient graph search have been studied extensively in Artificial Intelligence.

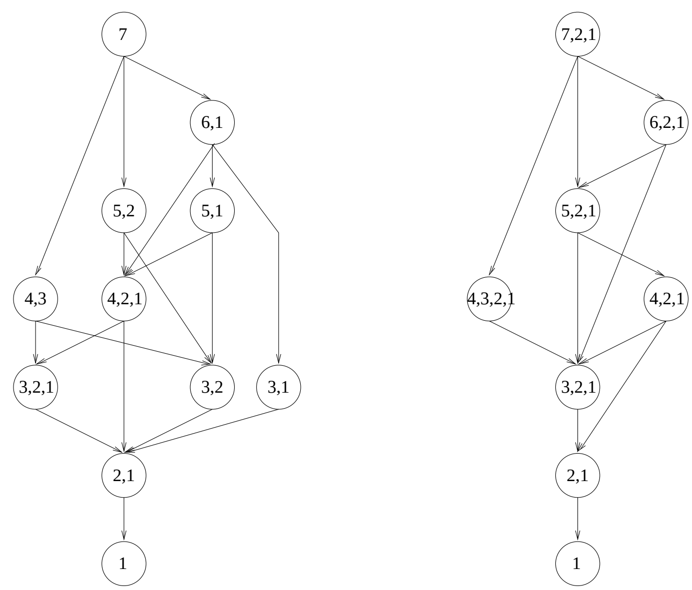


Figure 4.4: Search graphs for the sets $S = \{7\}$ and $S = \{7, 2, 1\}$.

These algorithms take great advantage of heuristic information. If $lb(S)$ is a function that is a lower bound on $\ell(S)$ for all sets $S$ and if $h(S)$ is the length of shortest path from the top node $\{n\}$ to the node $S$ in the search tree, then the length of the shortest addition chain for $n$ including $S$ is at least $h(S) + lb(S)$. A simple lower bound $lb(S)$ for a set $S$ that will be used throughout this section is the maximal length of an element in $S$.

$$lb(S) := max_{x \in S}(l_0(x)) \tag{4.7}$$

where we set $l_0(x) = \ell(x)$ if $\ell(x)$ is known and $l_0(x) = 0$ otherwise. More precise lower bounds will be discussed in Section 4.6.

One algorithm for a heuristic graph search is the $A^*$-Algorithm [Nil82, p.79]. This algorithm always expands the node whose evaluation function (i.e. $h(S) + lb(S)$) gives the lowest value among all nodes that are not yet expanded. The algorithm terminates when the goal node is found and all unexpanded nodes in the search tree have a higher or equal evaluation than the goal node. However, this algorithm requires a lot of memory as there are integers for which the computation of the shortest addition chain requires to examine a large number of nodes. For example the proof that $\ell(110591) > 21$ required to examine more than $8 \cdot 10^6$ nodes.

Therefore it seems natural to use a depth-first search algorithm. In order to avoid that the algorithm searches too deep we use a recursive procedure that searches an addition chain for a set $S$ having a maximal length $k$, starting with a small $k$ and increasing $k$ by 1 as long as no chain of length $k$ can be found.

**Search an addition chain of length $k$.**

Input: *Given a set $S$ and an upper bound $k$. This algorithm finds an addition chain of length $k$ or proves that no such chain exists.*

Step 1: *If $S = \{1\}$ and $k \geq 0$ then return the chain 1.*

Step 2: *If $lb(S) > k$ or if there is an entry in the hash table showing $\ell(S) > k$ then return 'no chain was found'.*

Step 3: *Let $m \leftarrow \max(S)$ be the largest element of $S$ and let $R \leftarrow S \setminus \{m\}$.*

Step 4: *Check for special cases*

Step 5: *Let $i \leftarrow m - 1$.*

Step 6: *If $\ell(i) > k - 1$ then set $i$ to the largest $j$ with $j < i$ and $\ell(j) \leq k - 1$.*

Step 7: *If $i < m/2$ then store $\ell(S) \geq k + 1$ in the hash table and return 'no chain was found'.*

Step 8: *Let $S' \leftarrow R \cup \{i, m - i\}$ and search an addition chain for $S'$ of length $k - 1$ at most by calling this procedure recursively. If a chain $C'$ for $S'$ is found then set $C \leftarrow C', m$ and return $C$.*

Step 9: *Set $i \leftarrow i - 1$ and goto step 6.*

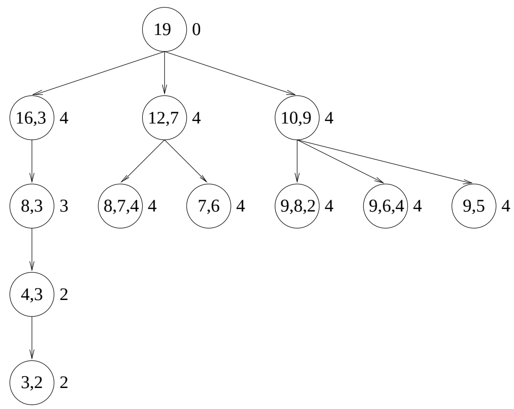


Figure 4.5: Search tree for the proof of $\ell(19) > 5$. The nodes of the search tree are labeled with the sets $S$ and the evaluation$h(S) + lb(S)$.

Step 4 is an optional step, which will be described in section 4.7. Step 6 asserts that the variable $i$ only runs over the integers $m > i \geq m/2$ that satisfy $\ell(i) < k-1$. This step is not necessary, but it can be implemented efficiently and results in a large speedup since the function $lb$ is much less frequently performed.

Hash tables are used to store lower bounds on the length of optimal addition chains for sets $S$. Only sets $S$ satisfying $lb(S) < \ell(S)$ are stored in those tables. The hash tables are not reinitialized before every computation of a new chain since lower bounds on sets $S$ can be used for the computation of $\ell(n)$ for different $n$'s. Therefore it is quite likely that the hash table overflows. The strategy that has been chosen to select the sets that are stored is quite simple. Only small sets (e.g. sets with no more than 4 or 5 elements) are stored, because it is not very likely that large sets occur more than once during the search and because small sets can be stored using less memory than large sets. In case of a hash collision the older entry is overwritten.

As an example Figure 4.5 shows the search tree for proving $\ell(19) > 5$ when the heuristic for computing lower bounds $lb$ defined in Equation 4.7 is used. This tree is remarkably small although $lb$ can be improved a lot. An algorithm that finds the shortest addition chain can now be implemented as follows.

**Computation of the length of the shortest addition chain of $n$.**

Input: $n$

Step 1: *Use heuristic methods for finding a short addition chain $C$ for $n$. Set $U$ to the length of $C$. Set $L$ to a lower bound for $\ell(c)$.*

Step 2: *If $U = L$ then return $U$.*

Step 3: *Use a recursive depth-first search to find an addition chain $C$ for $n$ of length exactly $L$ or prove that no such addition chain exists. If an addition chain for $n$ was found then return $L$ otherwise set $L \leftarrow L+1$ and go to step 2.*

When the lower bound $L$ for $n$ that is computed in step 1 of this algorithm is less than $\ell(n)-1$ then some nodes are computed more than once, since the algorithm proves that there is no addition chain of length $L$ first and that there is no addition chain for $L+1, \ldots$ in the following steps. However, it turns out that proving the nonexistence of an addition chain of length $x$ is generally much less time consuming than proving the nonexistence of an addition chain of length $x+1$. Contrary it may be a great waste of time if one tries to prove that $\ell(n) > x$ when an addition chain of length $x$ (or even a shorter one) exists.

It is important to have good heuristics for finding upper bounds. Trying to prove that $\ell(n) \geq k$ when $\ell(n) = k$ may be very time consuming. No such try is necessary when the heuristic for an upper bound has found an optimal chain. Heuristics for finding upper bounds are discussed in detail in Section 4.5.

An other advantage of our method is that it is very often possible to show that any short addition chain for a set $S$ must have a special form. For example an addition chain of length 7 for the set $\{29, 19\}$ cannot contain any integer $29 > x > 19$ since $\ell(19) = 6$ and implies that $\ell(\{29, x, 19\}) \geq 8$. Moreover, if it can be shown that different paths in the search chain lead to equivalent addition chains then it is necessary to follow only one path in the search graph. Improvements of this kind are discussed in Section 4.7.

## 4.5 Methods for finding upper bounds

The importance of having good upper bounds has been stated in Section 4.4. For example when we replaced a heuristic that found $\ell(n)$ for about 80 percent of all $n$'s by a heuristic that was successful in about 95 percent this resulted in a program that traversed only about one third of the nodes that the former program had traversed.

The basic strategy for finding upper bounds on $\ell(n)$ is to build addition chains using different heuristic methods and to chose the shortest one as an upper bound on $\ell(n)$. In this section we describe the heuristics that were used in the program.

**Theorem 4.7** *For all $m, n, k > 0$ we have*

$$\ell(n+1) \leq \ell(n) + 1 \tag{4.8}$$

$$\ell(n+2) \leq \ell(n) + 1 \text{ if } n \geq 2 \tag{4.9}$$

$$\ell(mn) \leq \ell(m) + \ell(n) \tag{4.10}$$

$$\ell(mn+k) \leq \ell(\{k, n\}) + \ell(m) + 1 \tag{4.11}$$

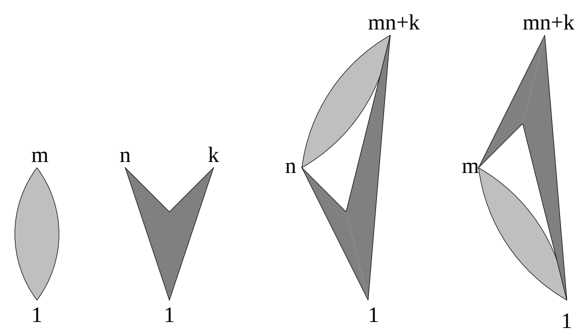


Figure 4.6: graphical representation of two chain constructions

$$\ell(\{mn+k,k,n\}) \leq \ell(\{k,n\}) + \ell(m) + 1 \tag{4.12}$$
$$\ell(\{mn+k,m\}) \leq \ell(\{k,n\}) + \ell(m) + 1 \tag{4.13}$$

*Proof:* Every addition chain $A = a_0, a_1, \ldots, a_r$ for $n \geq 2$ has $a_0 = 1$ and $a_1 = 2$. If $A$ is an optimal addition chain for $n$ then it follows that $A, n+1$ and $A, n+2$ are addition chains of length $\ell(n)+1$ for $n+1$ and $n+2$, respectively. Thus (4.8) and (4.9) are satisfied.

Let $A = 1, a_1, \ldots, a_{\ell(n)}$ and $B = 1, b_1, \ldots, b_{\ell(m)}$ be optimal addition chains for $n$ and $m$. An addition chain for $nm$ of length $\ell(n) + \ell(m)$ can then be constructed by

$$1, a_1, \ldots, a_{\ell(n)}, nb_1, \ldots, nb_{\ell(m)}.$$

This proves equation (4.10).

Let further $C = 1, c_1, \ldots, c_{\ell(\{k,n\})}$ be an optimal addition chain for the set $\{k, n\}$. An addition chain for $mn + k$ can then be constructed by possibly resorting the list

$$c_0, \ldots, c_{\ell(\{k,n\})}, nb_1, \ldots, nb_{\ell(m)}, mn + k.$$

It follows that (4.11) is satisfied. Moreover, since $k$ and $n$ are elements of the chain $C$ it follows that the more general equation (4.12) is satisfied too.

The graphical representation of this construction is shown in Figure 4.6. We assume that the two graphs shown on the left hand side of the figure represent optimal addition chains for the integer $m$ and the set $\{k, n\}$, respectively. In the first graph there are $m$ paths from the bottom vertex to the top vertex of the graph. In the second graph there exists $n$ paths from the bottom vertex to the vertex labeled $n$ and $k$ paths to the vertex labeled $k$. The third graph can be derived from these two graphs by matching the bottom vertex of graph 1 with vertex $n$ of graph 2 and matching vertex $m$ of graph 1 with vertex $k$ of graph 2. Counting the paths from the bottom vertex to the top vertex of the resulting graph shows that there are $mn$ paths through the matched node on the left and $k$ paths on the right side, thus resulting in $mn + k$ paths.

The fourth graph is the dual graph of the third graph. Thus it is a representation of an addition chain for $mn + k$ of length $\ell(\{n, k\}) + \ell(m)$. Since there are exactly $m$ paths from vertex 1 to vertex $m$ it follows that this graph is also a representation of an addition chain for the set $\{mn + k, k\}$. This shows that equation 4.13 is satisfied. □

Based on Theorem 4.7 the program tries to construct short addition chains. It should be noted here, that $\ell(k)$ is known for $k < n$ when the program is looking for an addition chain for $n$. The following methods are used to find an upper bound for $\ell(n)$:

- For $n \geq 2$ Equations (4.8) and (4.9) are used for a first upper bound.
- For all $i$ up to a bound $B_1$ (e.g. $B_1 = 100$) we use (4.10) or (4.11) depending on whether $i$ divides $n$. If $i$ is a divisor of $n$ then we have $\ell(n) \leq \ell(i) + \ell(n/i)$. This heuristic is called the factor method. Table 4.1 shows how successful this method is for different factors.

  If $i$ is not a divisor of $n$ then we have $\ell(n) \leq \ell(\{i, n \bmod i\}) + \ell(\lfloor n/i \rfloor) + 1$. Since both $i$ and $n \bmod i$ are less than $B_1$ it is possible to compute $\ell(\{i, j\})$ for $j < i \leq B_1$ once and store the result for later use.
- For all integers $i$ of the form $2^j$ or $2^j + 1$ with $i < n$ we use (4.10) or (4.11) depending on whether $i$ divides $n$. If $i$ is a divisor of $n$ then we have again $\ell(n) \leq \ell(i) + \ell(n/i)$. If $i$ does not divide $n$ then we have $\ell(n) \leq \ell(i) + \ell(\{\lfloor n/i \rfloor, n \bmod i\})$.

  The justification for this heuristic is that $\ell(i)$ is very small. However, it is necessary to find a good upper bound on $\ell(\{\lfloor n/i \rfloor, n \bmod i\})$. Good heuristics for this task have been proposed by Bos and Coster in [BC90].

## 4.6 Methods for finding lower bounds

One of the most time consuming part of the search is the proof that no addition chain for a given set and a given size exists. The search tree can be pruned a lot when good methods to prove lower bounds on $\ell(S)$ are available. It was our experience that even small looking improvements to the lower bound had often great effects on the running time of the program.

**Theorem 4.8** *Let $S$ be a set $\ell(S)$ the length of an optimal addition chain for $S$ and $m > 2^k \max(S)$ for $k > 0$. Then*

$$\ell(S \cup \{m\}) \geq \ell(S) + k + 1.$$

| $n$ | *percentage* | $n$ | *percentage* | $n$ | *percentage* |
|---|---|---|---|---|---|
| 2 | 99.5% | 31 | 11% | 65 | 86% |
| 3 | 78% | 33 | 75% | 67 | 24% |
| 5 | 66% | 37 | 22% | 71 | 1% |
| 7 | 33% | 41 | 31% | 73 | 32% |
| 11 | 16% | 43 | 30% | 77 | 34% |
| 13 | 32% | 47 | 5% | 79 | 3% |
| 17 | 70% | 49 | 54% | 83 | 46% |
| 19 | 16% | 53 | 7% | 89 | 5% |
| 23 | 33% | 59 | 14% | 97 | 70% |
| 29 | 7% | 61 | 16% | | |

Table 4.1: Percentage of integers $m$ with $\ell(mn) = \ell(m) + \ell(n)$ where $mn < 500{,}000$

*Proof:* Let $C = c_0, \ldots, c_r$ be an addition chain for the set $S \cup \{m\}$ and $n = \max(S)$. $n$ must be an element of the addition chain, say $n = c_j$. It follows $j \geq \ell(S)$ since otherwise we would have found an addition chain for $S$ that is shorter than $\ell(S)$. Since every element in an addition chain is at most twice as large as its predecessor it follows that $c_{j+k} \leq \max(S)2^k < m$. Thus there is no addition chain for $S \cup \{m\}$ of length $j + k$. $\Box$

Theorem 4.8 can be used to compute a lower bound on the length of an addition chain for a set $S$. We assume that $\ell(x)$ for all $x \in S$ is known.

**Compute a lower bound on the length of an addition chain for a set $S$.**

Input: *A nonempty set $S = \{s_0, \ldots, s_r\}$ where the integers $s_i$ are ordered i.e. $s_i < s_{i+1}$ for $0 \leq i < r$.*

Step 1: *Set $L_0 \leftarrow \ell(s_0)$ and $i \leftarrow 0$*

Step 2: *If $i = r$ then return $L_r$ otherwise set $i \leftarrow i + 1$.*

Step 3: *$k \leftarrow \lceil \log_2(s_i/s_{i-1}) \rceil - 1$, set $L_i \leftarrow \max(\ell(s_i), L_{i-1} + k + 1)$ and goto step 2*

*Remarks.* In that algorithm $L_i$ is a lower bound on the set $\{s_0, \ldots, s_i\}$ for all $0 \leq i \leq r$.

A few theorems have been found that can be used to determine $\ell(n)$ if $w(n)$ is small [Knu81, Theorem B and C, p. 449] [TC87]. Since the preceding algorithm finds $\ell(n)$ for such numbers very fast, it does not help much to use of these theorems here.

## 4.7 Transformations into equivalent chains

The concept of equivalent chains can be used to transform chains into other chains of equal length. This can often be used to prune large parts in the search tree. If it is possible to prove that any optimal addition chain in one part of the search tree can be transformed in an addition chain that will be found in an other subtree of the search tree, then it is not necessary to follow the former subtree, since we are interested in only one optimal addition chain. This section describes improvements of this form that have been used in the program.

**Theorem 4.9** *Let $x$ be an integer, $S$ be a set of integers and $\ell(S \cup \{x\}) = k$. Then at least one of the following conditions is true*

- *$x$ is even and $\ell(S \cup \{x/2\}) = k - 1$.*
- *There exists $1 \leq y < x/2$ such that $\ell(S \cup \{y, x - 2y\}) = k - 2$.*
- $\max(S) \cdot (k + 1 - \ell(S)) \geq x$.

*Proof:* Let $C$ be an addition chain for $\{x\} \cup S$, let $G = (V, E, \alpha, \omega)$ be the reduced graph for $C$, let $v$ be the vertex representing $x$ (i.e.$\mathrm{p}(w, v) = x$). Denote by $A = \{e_1, \ldots, e_r\} = \{e \in E : \omega(e) = v\}$ the set of vertices whose terminal node is $v$. Now select the vertex $v' \in \{\alpha(e) : e \in A\}$ that maximizes $y = \mathrm{p}w, v'$.

If $y \leq \max(S)$ then it follows that $r \max(S) \geq x$ and therefore $\max(S) \cdot (k + 1 - \ell(S)) \geq x$. On the other hand if $y > \max(S)$ then it follows that $A$ contains two edges $e_g \neq e_h$ with $\alpha(e_g) = \alpha(e_h) = v_i$ because $G$ is reduced and therefore every vertex not contained in $S \cup \{x\}$ must have out-degree $\geq 2$. Thus $x$ can be computed by

$$x = 2y + \sum_{\substack{1 \leq j \leq r \\ j \neq g, j \neq h}} \mathrm{p}(w, \alpha(e_j)).$$

Now either $A = \{e_g, e_h\}$ in which case we have $x = 2y$ and $\ell(\{y\} \cup S) = k - 1$ or $A \setminus \{e_g, e_h\} \neq \emptyset$. In the later case it follows that $\ell(S \cup \{x, y, x - 2y\}) = \ell(S \cup \{x\}) = k$ and therefore that we can find an addition chain for $S \cup \{x - 2y, y\}$ of length $k - 2$. $\square$

**Theorem 4.10** *Let $x$ be an integer and let $S$ be a set of integers with $2y \geq x > y > \max(S)$. If $\ell(x) = \ell(\{x, y\} \cup S) = k$ then $\ell(\{x - y, y\} \cup S) = k - 1$.*

*Proof:* Let $C$ be an addition chain for the set $\{x, y\} \cup S$ of length $k$. We have to show there exists an addition chain for the set $\{x - y, y\} \cup S$ of length $k - 1$. It follows from $\ell(x) = k$ that $y$ is needed to compute $x$ since otherwise we would get an addition chain for $x$ of length $k - 1$ by deleting $y$ in $C$. Since $y > x/2$ it follows that there exist integers $y = z_0, z_1, \ldots, z_r = x$ in $C$ such that the differences $w_i = z_{i+1} - z_i$ with $0 \leq i < r$ are contained in $C$. Since $z_1, \ldots, z_{r-1}$ are all greater than $y$ they are not contained in $S$. Therefore we may replace

$z_1, \ldots, z_{r-1}$ in $C$ by $w_0 + w_1, \ldots, w_0 + w_1 + \cdots + w_{r-1} = x - y$. After sorting the resulting list $C'$ we have again an addition chain for the set $\{x, y\} \cup S$ containing $x - y$. Since we have added as many elements to $C$ as we have deleted from this chain it follows that $C'$ is an addition chain for $\{x - y, y\} \cup S$ of length $k - 1$ at most. □

The following pseudo-code represents the improvements to the search algorithm. Step 4.1 is an immediate consequence of $S \subset T \Rightarrow \ell(S) \leq \ell(T)$. Step 4.2 is the implementation of Theorem 4.10 and Step 4.3 corresponds to Theorem 4.9.

**Check for special cases**

Step 4.1: *If there exist elements $x, y$ in $S'$ such that $x + y = m$ then search an addition chain for $S'$ of length $k - 1$. If such a chain $C$ is found then return $C, m$ otherwise return 'no chain was found'.*

Step 4.2: *If $2\max(S') > m$ and $\ell(x) = k$ then search an addition chain for $S' \cup \{x - y\}$ of length $k - 1$. If such a chain $C$ is found then return $C, m$ otherwise return 'no chain was found'.*

Step 4.3: *If $\max(R) \cdot (k + 1 - lb(R)) \geq x$ then goto Step 5.*

Step 4.3.1: *If $x$ is even then search an addition chain for $R \cup \{x/2\}$ of length $k - 1$. If such a chain $C$ is found then return the chain $C, x$.*

Step 4.3.2: *For all $1 \leq y < x/2$ search an addition chain for $R \cup \{y, x - 2y\}$ of length $k - 2$. If such a chain $C$ is found then return $C, x - y, x$.*

Step 4.3.3: *Return 'no chain was found'.*

## 4.8 Results

An implementation of the algorithm presented so far has been used to find shortests addition chains for all $n \leq 500{,}000$. This computation took about three weeks on a SPARC IPC. The running time for the computation up to $n$ is approximately $n^{\alpha(n)}$ where $\alpha(n)$ is a slowly increasing function. The measurements showed that $\alpha(n)$ is approximately 2 when $n$ is small and about 2.3 when $n$ is near $500{,}000$. However, the behavior of this function for larger $n$ is unknown.

The smallest $x$ which has length $n$ is denoted by $c(n)$. Table 4.2 shows the values of $c(n)$. This extends previously published results. Knuth [Knu81] lists $c$ up to $c(18) = 11231$ and Flammenkamp [Fla91] has computed $c$ up to $c(23) = 196591$.

Table 4.3 shows the number of addition chains of length $n$ denoted by $d(n)$. Our computations confirm the results of Flammenkamp up to $d(23)$. The emphasis of Flammenkamp's work was to find the numbers that have small addition chains (e.g. he extended Theorem B and C of [Knu81, section 4.6.3]). For this

| $n$ | $c(n)$ | $n$ | $c(n)$ |
|---|---|---|---|
| 1 | 2 | 13 | 607 |
| 2 | 3 | 14 | 1087 |
| 3 | 5 | 15 | 1903 |
| 4 | 7 | 16 | 3583 |
| 5 | 11 | 17 | 6271 |
| 6 | 19 | 18 | 11231 |
| 7 | 29 | 19 | 18287 |
| 8 | 47 | 20 | 34303 |
| 9 | 71 | 21 | 65131 |
| 10 | 127 | 22 | 110591 |
| 11 | 191 | 23 | 196591 |
| 12 | 379 | 24 | 357887 |

Table 4.2: $c(n)$ is the least $i$ with $\ell(i) = n$

reason he achieved to find even $d(24)$. This is the main difference to the algorithm presented in this chapter, since it was our goal to find an algorithm that computes $c(n)$ as efficiently.

The values for $m$ that satisfy $\ell(3m) < \ell(m)$ are shown in table 4.5. No other value $mn \leq 500,000$ satisfies $\ell(mn) < \ell(m)$.

| $n$ | $d(n)$ | $n$ | $d(n)$ |
|---|---|---|---|
| 1 | 1 | 11 | 246 |
| 2 | 2 | 12 | 432 |
| 3 | 3 | 13 | 772 |
| 4 | 5 | 14 | 1382 |
| 5 | 9 | 15 | 2481 |
| 6 | 15 | 16 | 4490 |
| 7 | 26 | 17 | 8170 |
| 8 | 44 | 18 | 14866 |
| 9 | 78 | 19 | 27128 |
| 10 | 136 | 20 | 49544 |

Table 4.3: $d(n) = \text{card}\{x : \ell(x) = n\}$

| $n$ | $m$ |
|---|---|
| 2 | 191 |
| 3 | 171 |
| 5 | 3277 |
| 6 | 2731 |
| 7 | 28087 |
| 9 | $\leq 233017$ |

Table 4.4: Smallest $m$ with $\ell(m) = \ell(mn)$

| | | |
|---|---|---|
| 2731 | 43691 | 87392 |
| 5462 | 43692 | 92843 |
| 10923 | 43696 | 109228 |
| 10924 | 54614 | 109230 |
| 13655 | 54615 | 109240 |
| 21846 | 54620 | 133819 |
| 21848 | 71003 | 142006 |
| 27307 | 87382 | 152919 |
| 27310 | 87384 | |

Table 4.5: The set of values $m \leq 166'666$ satisfying $\ell(3m) < \ell(m)$.

# Chapter 5

# Lucas chains

In chapter 4 we have studied the computation of a sequence $x_1, \ldots$ that satisfied a relation of the form $x_{i+j} = f(x_i, x_j)$ for a function $f$ that is efficiently computable. In this chapter we will study sequences that satisfy a relation of the form $x_{i+j} = f(x_i, x_j, x_{i-j})$. Given $n, x_0$ and $x_1$ we are looking for the smallest number of evaluations of $f$ to compute $x_n$. Examples of functions satisfying such a relation are $V_n(P,1)$ and $cos(n)$, since these functions satisfy

$$\begin{aligned} V_{n+m}(P,1) &= V_n(P,1)V_m(P,1) - V_{n-m}(P,1) \\ \cos(n+m) &= 2\cos(n)\cos(m) - cos(n-m) \end{aligned}$$

Such recurrences also appear in the computation of elliptic curves in certain parametrizations [Mon87, p.260–261].

**Definition 5.1** *A* Lucas chain *for a positive integer $n$ is an ascending list of integers $a_{-1}, a_0, \ldots, a_r$ such that $a_{-1} = 0, a_0 = 1$, $a_r = n$ and such that for all $1 \le k \le r$ there exists $0 \le h, i, j < k$ with $a_k = a_i + a_j$ and $a_h = a_i - a_j$. The length of this chain is $r$. The minimal length of a Lucas chain for $n$ is denoted by $\ell_L(n)$. A Lucas chain for a set $S$ of nonnegative integers is a Lucas chain containing all elements of $S$. The minimal length of a Lucas chain for $S$ is denoted by $\ell_L(S)$.*

*Example:* A Lucas chain of length 7 is $0, 1, 2, 3, 4, 5, 9, 14, 23$. Thus we may compute $V_{23}(x,1)$ by

$$\begin{aligned} V_0(x,1) &= 2 \\ V_1(x,1) &= x \\ V_i(x,1) &= V_{i-1}(x,1) \cdot V_1(x,1) - V_{i-2}(x,1) \quad \text{for } i = 2,3,4,5 \\ V_9(x,1) &= V_5(x,1) \cdot V_4(x,1) - V_1(x,1) \\ V_{14}(x,1) &= V_9(x,1) \cdot V_5(x,1) - V_4(x,1) \\ V_{23}(x,1) &= V_{14}(x,1) \cdot V_9(x,1) - V_5(x,1) \end{aligned}$$

## 5.1 Binary Method

A first method to compute Lucas chains is the *binary method.* This method finds a reasonably short Lucas chains $B_L(n-1,n)$ for the sets $\{n, n-1\}$. A recursive description of this method is given by the following equations.

$$B_L(1,2) = 0,1,2 \tag{5.1}$$
$$B_L(2,3) = 0,1,2,3 \tag{5.2}$$
$$B_L(2n,2n+1) = B_L(n,n+1),2n,2n+1 \tag{5.3}$$
$$B_L(2n-1,2n) = B_L(n-1,n),2n-1,2n \tag{5.4}$$

This method shows that a Lucas chain of length at most $2\log_2(n)$ exists.

## 5.2 An algorithm for finding optimal Lucas chains

This section describes the outline of an algorithm to find the length of optimal Lucas chains. A first implementation of an algorithm that computed $\ell_L(x)$ directly through a graph search proved to be inefficient. However, we will show that the sequence $\ell_L(x) : 1 \le x \le B$ can be efficiently computed from another sequence $\widehat{\ell_L}(x) : 1 \le x \le B$. It turned out that the computation of this sequence is a lot faster than the direct computation of $\ell_L(x)$.

**Definition 5.2** *Let $C = 0, 1, s_1, \ldots, s_r$ be a Lucas chain. A* doubling step *denotes two consecutive elements $s_i, s_{i+1}$ where $s_{i+1} = 2s_i$. A Lucas chain without doubling step $s_{i+1} = 2s_i$ for $i \ge 1$ is called* simple. *A Lucas chain is called* composite *if it is of the form*

$$0, 1, a_1, a_2, \ldots, a_r, b_1 a_r, b_2 a_r, \ldots, b_s a_r$$

*where $0, 1, a_1, a_2, \ldots, a_r$ and $0, 1, b_1, b_2, \ldots, b_s$ are Lucas chains. For a set of integers $S$ we will denote by $\widehat{\ell_L}(S)$ the length of the shortest simple Lucas chain for $S$. A Lucas chain $C = 0, 1, s_1, \ldots, s_r$ for a set $S$ will be called* degenerate *if one element in the chain is not necessary, i.e. there exists $1 \le i \le r$ such that $0, 1, s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_r$ is a Lucas chain for $S$.*

*Remark.* A composite Lucas chain can always be transformed into a simple one by inserting a new element between two elements of a doubling step. It is therefore possible to define alternatively $\widehat{\ell_L}(S)$ as the minimal sum of the length and the number of doubling steps $s_i, s_{i+1}$ with $i > 1$ of an addition chain. Table 5.1 shows some optimal chains. An optimal simple Lucas chains $n$ may be degenerate. For example the simple Lucas chain for $n = 10$ in Table 5.1 is degenerate, since the number 8 is not necessary.

We will see in the sequel that the function $\widehat{\ell_L}(S)$ is very useful for different reasons.

| $n$ | $\ell_L(n)$ | Lucas chain for | $\widehat{\ell_L}(n)$ | simple Lucas chain for $n$ |
|---|---|---|---|---|
| 2 | 1 | $0,1,2$ | 1 | $0,1,2$ |
| 3 | 2 | $0,1,2,3$ | 2 | $0,1,2,3$ |
| 4 | 2 | $0,1,2,4$ | 2 | $0,1,2,3,4$ |
| 5 | 3 | $0,1,2,3,5$ | 3 | $0,1,2,3,5$ |
| 6 | 3 | $0,1,2,3,6$ | 4 | $0,1,2,3,4,6$ |
| 7 | 4 | $0,1,2,3,4,7$ | 4 | $0,1,2,3,4,7$ |
| 8 | 3 | $0,1,2,4,8$ | 4 | $0,1,2,3,5,8$ |
| 9 | 4 | $0,1,2,3,6,9$ | 5 | $0,1,2,3,4,5,9$ |
| 10 | 4 | $0,1,2,3,5,10$ | 5 | $0,1,2,3,5,8,10$ |

Table 5.1: Optimal Lucas chains and simple Lucas chains for $n \leq 10$.

- The sequence $\ell_L(x) : 1 \leq x \leq N$ can be computed from the list $\widehat{\ell_L}(x) : 1 \leq x \leq N$.
- It seems easier to find a good lower bound on $\widehat{\ell_L}(S)$ than to find a good lower bound on $\ell_L(S)$.
- The algorithm searching for $\widehat{\ell_L}(x)$ proved to be much faster than the algorithm searching for $\ell_L(x)$ directly.

Montgomery observed that doubling steps do not occur very often in Lucas chains. This observation is based on Theorem 5 of [Mon].

**Theorem 5.3 (Montgomery)** *If $s_{i+1} = 2s_i$ in a Lucas chain then $s_i$ divides $s_j$ for all $j \geq i$.*

An immediate consequence of this theorem is

**Corollary 5.4** *A Lucas chain is either simple or composite.*

It follows that $\ell_L(x)$ either is equal to $\widehat{\ell_L}(x)$ or there exists a divisor $d$ of $x$ such that $\ell_L(x) = \ell_L(d) + \ell_L(x/d)$.

An algorithm that computes the sequence $\ell_L(x) : 1 \leq x \leq B$ from $\widehat{\ell_L}(x) : 1 \leq x \leq B$ can therefore be described as follows.

**Computation of $\ell_L(x)$ from $\widehat{\ell_L}(x)$.**

Step 1: *for all* $1 \leq i \leq B$ *set* $L[i] \leftarrow \widehat{\ell_L}(i)$

Step 2: *Set* $i \leftarrow 2$.

Step 3: *Set* $j \leftarrow i$.

Step 4: *Set* $L[ij] \leftarrow \min(L[ij], L[i] + L[j])$

Step 5: *Set* $j \leftarrow j+1$. *If* $ij \leq B$ *then goto Step 4.*
Step 6: *Set* $i \leftarrow i+1$. *If* $i^2 \leq B$ *then goto Step 3.*
Step 7: *Now return the Array* $L$ *as the result for* $\ell_L$ *(i.e* $\ell_L(i) = L[i]$*).*

The definition of *simple* may look cumbersome at first sight. An important reason not to exclude degenerate chains from the definition of simple Lucas chains is the following property.

$$S \subseteq T \Rightarrow \widehat{\ell_L}(S) \leq \widehat{\ell_L}(T) \tag{5.5}$$

This can easily be verified, since every simple Lucas chain for the set $T$ is a simple Lucas chain for $S$ too. However, an analogue equation for simple nondegenerate Lucas chains is not true. For example

$$0, 1, 2, 3, 4, 6, 10, 16, 26, 42$$

is a simple nondegenerate Lucas chain of length 8 for the set $\{3, 16, 26, 42\}$ but not for the set $\{16, 26, 42\}$ since the element 3 is now superfluous. The shortest simple nondegenerate Lucas chain for $\{16, 26, 42\}$ is

$$0, 1, 2, 3, 5, 8, 13, 16, 26, 29, 42,$$

which is a chain of length 9.

We will search an optimal simple Lucas chain by calling two algorithms, one searching for simple degenerate Lucas chain and one searching for simple nondegenerate Lucas chains.

If a simple degenerate Lucas chain for $n$ has length $\widehat{\ell_L}(n)$ then this implies that the the chain becomes composite when the superfluous (or one of the superfluous) element is removed from the chain. We can therefore describe the algorithm for searching simple degenerate Lucas chains as follows.

**Search a simple degenerate Lucas chain of length $k$**

Input: *Given an integer* $n$ *and an upper bound* $k$. *This algorithm finds a simple degenerate Lucas chain for* $n$ *of length* $k$ *or proves that no such chain exists.*

Step 1: *If a divisor* $1 < d < n$ *of* $n$ *exists such that* $\widehat{\ell_L}(d) + \widehat{\ell_L}(n/d) + 1 \leq k$ *then search an optimal simple Lucas chain* $A = 0, 1, a_1, \ldots, a_{\widehat{\ell_L}(d)}$ *for* $d$ *and an optimal simple Lucas chain* $B = 0, 1, b_1, \ldots, b_{\widehat{\ell_L}(n/d)}$ *for* $n/d$. *Find* $i, j \geq 0$ *with* $a_i + a_j = a_{\widehat{\ell_L}(d)}$. *Return the Lucas chain*

$$0, 1, a_1, \ldots, a_{\widehat{\ell_L}(d)}, a_{\widehat{\ell_L}(d)} + a_i, b_1 d, \ldots, b_{\widehat{\ell_L}(n/d)} d.$$

*Otherwise return 'no such chain was found'.*

Simple nondegenerate Lucas chains can be found by a recursive algorithm that is similar to the one used for addition chains. The algorithm uses the function $lb(S)$ which computes a lower bound on $\widehat{\ell}_L(S)$. This function will be examined in detail in Section 5.4.

**Search a simple nondegenerate Lucas chain of length $k$.**

Input: *Given a set $S$ and an upper bound $k$. This algorithm finds a simple nondegenerate chain of length $k$ or proves that no such chain exists.*

Step 1: *If $\max(S) = 1$ and $k \geq 0$ then return the chain $0, 1$. If $\max(S) = 2$ and $k \geq 1$ then return the chain $0, 1, 2$.*

Step 2: *If $lb(S) > k$ then return 'no chain was found'.*

Step 3: *Let $m \leftarrow \max(S)$ be the largest element of $S$ and let $R \leftarrow S \setminus \{m\}$.*

Step 4: *Check for special cases*

Step 5: *If $R = \emptyset$ or $2\max(R) \leq m$ then set $u \leftarrow \lfloor m/2 \rfloor + 1$ otherwise set $u \leftarrow \lceil m/2 \rceil$.*

Step 6: *Let $i \leftarrow m - 1$.*

Step 7: *If $\ell_L(i) > k - 1$ then set $i$ to the largest $j$ satisfying $j < i$ and $\ell_L(j) \leq k - 1$.*

Step 8: *If $i < u$ return 'no chain was found'.*

Step 9: *Let $S' \leftarrow R \cup \{i, m - i, i - (m - i)\}$ and search an simple nondegenerate Lucas chain for $S'$ of length $k - 1$ at most by calling this procedure recursively. If a chain $C'$ for $S'$ is found then set $C \leftarrow C', m$ and return $C$.*

Step 10: *Set $i \leftarrow i - 1$ and goto step 7.*

*Remarks.* Step 5 ensures that $i = m/2$ is tried if and only if there exists an element in $S$ that is larger than $m/2$, since otherwise the resulting chain would not be simple.

## 5.3 Upper bounds on the length of Lucas chains

A variant of the algorithms `PRAC` of [Mon] was used to compute upper bounds on $\widehat{\ell}_L(x)$. This algorithm tries to find short simple Lucas chains for sets $\{x, y, x - y\}$ where $\gcd(x, y) = 1$ and $y \approx x\phi$ where $\phi = (\sqrt{5} - 1)/2 \approx 0.618$. Table 5.2 shows the rules that have been used to generate a Lucas chain for such a set. The algorithm always selects the first rules whose condition is satisfied.
*Remark.* The algorithm `PRAC` of [Mon] is more sophisticated than our version. The only reason why this algorithm had not been implemented was that our version was good enough to find an optimal chain in most cases.

| Nr. | Condition | Chain |
|---|---|---|
| 1. | $x = 3$ | $0, 1, 2, 3$ |
| 2. | $z > y$ | $P(x, z, y)$ |
| 3. | $4z \geq y$ | $P(y, z, y - z), x$ |
| 4. | $x \equiv 2 \pmod 2$ | $P(x/2, x - y, z), y, x$ |
| 5. | $y \equiv 2 \pmod 2$ | $P(x - y, y/2, z), y, x$ |
| 6. | $z \equiv 2 \pmod 2$ | $P((x + y)/2, y, z/2), z, x$ |

Table 5.2: A set of rules to compute a Lucas chain $P(x, y, z)$ for a set $\{x, y, z\}$ where $x = y + z$ and $\gcd(x, y) = 1$.

## 5.4 Lower bounds on the length of Lucas chains

The main purpose of computing lower bounds is to speed up the computation of optimal chains. Since computing $\ell_L(n)$ via $\widehat{\ell_L}(n)$ is much more efficient only lower bounds on $\widehat{\ell_L}(n)$ and $\widehat{\ell_L}(S)$ will be considered in this section.

A very good lower bound on $\widehat{\ell_L}(n)$ is given by Theorem 8 of [Mon].

**Theorem 5.5** *Let $r = \widehat{\ell_L}(n)$. Then*

*(i)* $n \leq F_{r+2}$

*(ii) If* $n \neq F_{r+2}$ *then* $n \leq F_{r+2} - F_{r-3}$

*(iii) If* $n > L_r$, *then* $|5n - 2L_{r+2}|$ *is a Lucas number.*

$F_n$ and $L_n$ denote the $n$-th Fibonacci and Lucas number (i.e. $F_n = U_n(1, -1)$ and $L_n = V_n(1, -1)$). Montgomery proves that these bounds are valid for Lucas chains satisfying $\ell_L(x) < \ell_L(d) + \ell_L(x/d)$ for all divisors $1 < d < x$ of $x$. In the proof it is assumed only that the corresponding Lucas chain is simple. Therefore his proof is also valid for Theorem 5.5. An immediate consequence of this theorem is the following algorithm.

**Compute a lower bound on the length of a simple Lucas chain for a set $S$.**

Input: *A nonempty set $S = \{s_0, \ldots, s_k\}$ where the integers $s_i$ are ordered i.e. $s_i < s_{i+1}$ for $0 \leq i < r$.*

Step 1: *Set $j \leftarrow 0, a_j \leftarrow 1, b_j \leftarrow 1$ and $i \leftarrow 0$.*

Step 2: *While $a_j < s_i$ set $a_{j+1} \leftarrow a_j + b_j$, $b_{j+1} \leftarrow a_j$ and $j \leftarrow j + 1$.*

Step 3: *If $j < \widehat{\ell_L}(s_i)$ then set $j \leftarrow \widehat{\ell_L}(s_i)$ and set $b_j \leftarrow x$ where $x$ is the largest $x < s_i$ with $\widehat{\ell_L}(x) < \widehat{\ell_L}(s_i)$.*

Step 4: *Set $a_j \leftarrow s_i$ and $i \leftarrow i + 1$. If $i \leq k$ then goto Step 2.*

Step 5: *Return $j$ as a lower bound for $\widehat{\ell_L}(S)$.*

*Remarks.* If $0, 1, c_1, \ldots, c_r$ is a simple Lucas chain for $S$ then for all $0 \le j \le r$ the following two invariants $c_j \le a_j$ and $c_{j-1} \le b_j$ are satisfied.

## 5.5 Special cases

Our algorithm considers two special cases for simple nondegenerate Lucas chains. If the maximal element $m$ of a set of integers $S$ can be computed from the remaining elements $R = S \setminus \{m\}$ and if $2\max(R) > m$ then there exists a simple nondegenerate Lucas chain for $S$ of length $k$ if and only if a simple nondegenerate Lucas chain for $R$ of length $k-1$ exists.

**Check for special cases**

Step 4.1 *If $m$ is even and $m/2 \in R$ and $\max(R) > x$ then search a simple nondegenerate Lucas chain $C'$ of length $k-1$ for the set $R$. If such a chain is found then return $C', x$ otherwise return 'no chain was found'.*

Step 4.2 *If there exists positive integers $x, y, z \in R$ such that $z = x - y$ and $x + y = m$ then search an simple nondegenerate Lucas chain $C'$ of length $k-1$ for the set $R$. If such a chain is found then return $C', x$ otherwise return 'no chain was found'.*

## 5.6 Results

Table 5.6 show the smallest number that has an optimal Lucas chain for a given length.

The first example satisfying $\ell_L(m) = \ell_L(2m)$ is $m = 7193$, since then $\ell_L(7193) = \ell_L(14386) = 20$. An optimal chain for 14386 is 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 843, 1453, 2296, 3749, 6045, 8341, 14386.

It is remarkable, that Theorem 8 of [Mon] predicts the correct lower bound for 2194 out of 3245 primes less than 30,000. The lower bounds for Lucas chains seem to be much closer to $\ell_L(n)$ than any known lower bound for addition chains. The factor method is more often correct for Lucas chains than for addition chains. The frequency for $\ell_L(mn) < \ell_L(m) + \ell_L(n)$ for different primes $n$ is shown in Table 5.4.

| $n$ | $c_l(n)$ | $n$ | $c_l(n)$ | $n$ | $c_l(n)$ |
|---|---|---|---|---|---|
| 1 | 2 | 11 | 127 | 21 | 10973 |
| 2 | 3 | 12 | 197 | 22 | 17539 |
| 3 | 5 | 13 | 331 | 23 | 26627 |
| 4 | 7 | 14 | 461 | 24 | 43103 |
| 5 | 11 | 15 | 739 | 25 | 67079 |
| 6 | 17 | 16 | 1123 | 26 | 109639 |
| 7 | 23 | 17 | 1801 | 27 | 160649 |
| 8 | 37 | 18 | 2903 | 28 | 271829 |
| 9 | 53 | 19 | 4463 | 29 | 432199 |
| 10 | 83 | 20 | 6947 | | |

Table 5.3: The smallest number $c_l(n)$ with a Lucas chain of length $n$

| $n$ | *percentage* | $n$ | *percentage* | $n$ | *percentage* |
|---|---|---|---|---|---|
| 2 | 0.02% | 29 | 0.6% | 67 | 1.8% |
| 3 | 0.20% | 31 | 0.1% | 71 | 0.9% |
| 5 | 0.14% | 37 | 14.4% | 73 | 0.6% |
| 7 | 1.30% | 41 | 5.6% | 79 | 0.03% |
| 11 | 1.31% | 43 | 1.5% | 83 | 34.5% |
| 13 | 0.03% | 47 | 0.3% | 89 | 0.0% |
| 17 | 2.38% | 53 | 31.9% | 97 | 7.3% |
| 19 | 0.19% | 59 | 12.4% | | |
| 23 | 17.42% | 61 | 8.2% | | |

Table 5.4: Percentage of integers $n$ with $\ell_L(mn) < \ell_L(m) + \ell_L(n)$ where $mn < 500,000$

# Chapter 6

# Analysis of some number-theoretic public-key cryptosystems

The first part of this chapter is devoted to ElGamal's signature scheme [Gam85]. This scheme relies on the difficulty of computing discrete logarithm in the multiplicative group $\mathbb{F}_p^*$ and can therefore be broken if the computation of discrete logarithms is feasible. However, the converse has never been proved. In the first section we show that it is sometimes possible to forge signatures without breaking the underlying discrete logarithm problem. This shows that the ElGamal signature scheme and some variants of the scheme must be used very carefully.

Section 6.1.1 describes the ElGamal signature scheme. In Section 6.1.2 we present a method to forge signatures if some additional information on the generator is known. We will show that signatures can be forged if the generator $\alpha$ is smooth and divides $p-1$. Hence for example $\alpha = 2$ is an insecure choice. In Section 6.1.3 we discuss the case where the public parameters are not chosen by the user itself. The authority that chooses these parameters may generate some trapdoor information, which will allow it to sign arbitrary messages for any user. Section 6.1.4 contains a brief description of some possible countermeasures. In Section 6.1.5 we discuss a variant of the ElGamal scheme over $\mathbb{Z}/(n)$ with $n = pq$, which was believed to be as hard as factoring and computing discrete logarithms. However, the factorization of $n$ can often be found from known signatures. Moreover, we will show that in this case computation in only one of the groups $\mathbb{F}_p^*$ or $\mathbb{F}_q^*$ is sufficient to forge signatures. Again it will not be necessary to discover the complete secret key of a user to generate signatures.

The second part of the chapter is devoted to Lucas-based RSA. A cryptosystem based on Lucas functions was proposed in [MN81]. It has been claimed in [Smi93] that no chosen message attack for Lucas-based RSA is possible. However, it is shown here that such an attack is possible.

## 6.1 ElGamal's signatures scheme

### 6.1.1 Description of the scheme

ElGamal's signature scheme can be described as follows.

**Public parameters.** Let $p$ be a large prime and $\alpha$ a generator of the multiplicative group $\mathbb{F}_p^*$. These parameters are not part of the (secret key, public key) pair and can be shared among different users.

**Secret key and public key.** Every user $A$ chooses a secret key $x_A \in \{0, \ldots, p-2\}$ and publishes $y_A \equiv \alpha^{x_A} \pmod{p}$ as his public key.

**Computation of a signature.** Let $0 \leq h < p-1$. To compute a signature on $h$ user $A$ proceeds as follows. First he chooses a random value $k \in \{0, \ldots, p-2\}$ relatively prime to $p-1$ and computes $1 \leq r < p$ and $s$ by

$$\begin{array}{rcl} r & \equiv & \alpha^k \pmod{p} \\ \text{and} \quad s & \equiv & (h - x_A r)k^{-1} \pmod{p-1}. \end{array}$$

The pair $(r, s)$ is a valid signature on $h$.

**Verification of a signature.** Any user knowing the public key $y_A$ can verify the signature by checking that $1 \leq r < p$ and the following equation are satisfied.

$$\alpha^h \equiv r^s y_A^r \pmod{p}$$

The ElGamal signature scheme can be broken when discrete logarithms in $\mathbb{F}_p^*$ can be computed. The prime $p$ must therefore be chosen large enough to prevent the computation of discrete logarithms by the number field sieve [Gor93] and $p-1$ must contain at least one large prime factor to disable the algorithm of Pohlig and Hellman [PH78]. The value $h$ that occurs in the signature is normally not equal to the message to sign, it is rather the result of a collision free hash function applied to the message. This avoids the existential attack described in [Gam85]. It is important that the verifier checks whether $1 \leq r < p$ is satisfied. If he would accept signatures where $r$ is larger than $p$ then any signature $(r, s)$ on $h$ could be used to generate a signature $(r_2, s_2)$ on an arbitrary hash value $h_2$ by setting $u \equiv h_2 h^{-1} \pmod{p-1}$. This implies

$$\alpha^{h_2} \equiv \alpha^{hu} \equiv (y_A)^{ru} r^{su} \pmod{p}.$$

Now $(r_2, s_2)$ can be found by setting $s_2 \equiv su \pmod{p-1}$ and computing $r_2$ satisfying $r_2 \equiv ru \pmod{p-1}$ and $r_2 \equiv r \pmod{p}$ by using the Chinese Remainder Theorem. This kind of attack will for example be used in Section 6.1.5.

### 6.1.2 Weak generators

The security of the ElGamal signature scheme depends heavily on the parameters $p$ and $\alpha$. The following theorem shows that ElGamal signatures can be generated when some additional information on the generator $\alpha$ is available.

**Theorem 6.1** *Let $p-1=bw$ where $b$ is smooth and let $y_A \equiv g^{x_A} \pmod{p}$ be the public key of user $A$. If $r$ and $k$ are known, such that $r \equiv \alpha^k \equiv cw \pmod{p}$ with $0<c<b$ then it is possible to generate a valid ElGamal signature $(r,s)$ for all $h$ with $h \equiv x_A r \pmod{\gcd(k,p-1)}$ can be found. In particular when $r$ is a generator of $\mathbb{F}_p^*$ then it is possible to generate an ElGamal signature for all $h$.*

*Proof:* First we show that the equation

$$\alpha^{wz} \equiv (y_A)^w \pmod{p}$$

can be solved for $z$. It follows from $p-1=bw$ that the subgroup $H$ generated by $\alpha^w$ has order $b$. Since $b$ is smooth it is possible to compute discrete logarithms in $H$ by using the algorithm of Pohlig and Hellman [PH78], so that $z$ can be found. Now let

$$\begin{aligned} f &= \gcd(k,p-1) \\ s &\equiv \frac{h-cwz}{f}\left(\frac{k}{f}\right)^{-1} \pmod{(p-1)/f}. \end{aligned}$$

It is important that $f$ divides $h-cwz$ and this condition is satisfied if and only if $h \equiv x_A r \pmod{\gcd(k,p-1)}$. We will show that $(r,s)$ is a valid signature on $h$. We have

$$sk \equiv h - cwz \pmod{p-1}$$

and therefore

$$\begin{aligned} y_A^r r^s &\equiv \alpha^{sk}(y_A)^{cw} \\ &\equiv \alpha^{h-cwz}\alpha^{cwz} \equiv \alpha^h \pmod{p}. \quad \Box \end{aligned}$$

**Corollary 6.2** *If $\alpha$ is smooth and divides $p-1$ then it is possible to generate a valid ElGamal signature on an arbitrary value $h$ if $p \equiv 1 \pmod{4}$ and on one half of the values $0 \le h < p$ if $p \equiv 3 \pmod{4}$.*

*Proof:* Let $k=(p-3)/2$. Then

$$\alpha^k \equiv \alpha^{(p-1)/2}\alpha^{-1} \equiv (-1)\alpha^{-1} = (p-1)/\alpha \pmod{p}.$$

Thus it follows from Theorem 6.1 that valid signatures can be generated when $h \equiv x_A r \pmod{\gcd(k,p-1)}$. If $p \equiv 1 \pmod{4}$ then $\gcd((p-3)/2, p-1) = 1$ and thus signatures for all $h$ can be generated. If $p \equiv 3 \pmod{4}$ then $\gcd((p-$

$3)/2, p-1) = 2$. This implies that signatures for one half of all $h$'s can be generated. □

Thus if the generator $\alpha$ is chosen badly then signatures on every given message can be found without knowing the secret key. Choosing $\alpha = 2$ is exceptionally bad since it allows to forge signatures for all odd primes $p$ and at least one half of all messages $h$. Such small generators are sometimes chosen in order to get an efficient exponentiation.

It should be noted that this attack succeeds because of the special choice of $r$, which reduces the discrete logarithm problem in $\mathbb{F}_p^*$ to the discrete logarithm problem in a subgroup of $\mathbb{F}_p^*$ with smooth order. Another attack that is based on the fact that discrete logarithms in small subgroups are computable has been described recently by Menezes Qu and Vanstone [MQV95]. (Their attack shows that authenticity is not guaranteed in a few Diffie-Hellman based key agreement protocols.)

### 6.1.3 Constructing a trapdoor

The public parameters $p$ and $\alpha$ may be fixed so that every user in the system uses the same group and generator. Such a convention is attractive because it allows the use of shorter public keys. Additionally, signatures can be computed and verified much faster by using precomputed exponents [BGMW93]. However, the authority choosing the prime $p$ and the generator $\alpha$ for a signature system may additionally generate some trapdoor information that can be used to forge arbitrary messages later.

One way to generate a prime $p$ such that $\mathbb{F}_p^*$ has a trapdoor has been pointed out in [BFSe92, p.50]. The prime $p$ can be generated together with a polynomial that is highly suitable for the number field sieve. The authority will then be able to compute discrete logarithms faster than a user who does not know the trapdoor [Gor92]. However, this gives only a moderate advantage and can be avoided by chosing the prime $p$ sufficiently large. Moreover, such primes can be recognized fairly easily [RLS+93].

Here we present another way to generate a trapdoor. An authority that can choose the public parameters $p$ and $\alpha$ can generate these parameters such that it additionally knows secret values $r$ and $k$ satisfying the constraints of Theorem 6.1. We illustrate this possibility by giving two different methods to generate the trapdoor. The first method shows how a generator $\alpha$ and the trapdoor information can be generated given a fixed prime $p$. The second method shows how a prime $p$ and the trapdoor information can be generated given a fixed small generator $\alpha$. There is no guarantee that these methods succeed. However, the probability of success is sufficiently high to threaten the security of the system.

**Method A.** Let $p - 1 = bw$ with $b$ smooth. If $b$ is not too small then we can find $\alpha, r$ and $k$ in the following way. We choose $c \in \{1, \ldots, b-1\}$ randomly until $r = cw$ is a generator of $\mathbb{F}_p^*$. Finally we chose $k$ with $\gcd(k, p-1) = 1$ and compute $\alpha = r^{k^{-1}}$.

This attack is practical when a generator $r = cw$ of $\mathbb{F}_p^*$ can be found in reasonable time. When $b$ is too small then no generator of the form $cw$ may exist. However, the number of generators of $\mathbb{F}_p^*$ is $\varphi(p-1)$. Since $n/\varphi(n) = O(\ln\ln(n))$ (see for example [HW79]) we expect to find a generator after $O(\ln\ln(p))$ trials.

When $k$ is chosen uniformly from the set of all $1 \le k < p-1$ that are relatively prime to $p-1$ then $\alpha = r^{k^{-1}}$ is a random generator of $\mathbb{F}_p^*$. Thus it is impossible to detect the trapdoor as long as no false signature has been published. However, the trapdoor can be reconstructed from a given false signature $(r, s)$ on $h$ since $\alpha^h \equiv y_A^r r^s \pmod p$ implies $h \equiv ks \pmod w$. In order to find $k$ it is then sufficient to compute $\log_\alpha(r) \pmod b$ and this can be done efficiently as $b$ is smooth.

**Method B.** When the generator $\alpha$ is fixed then $p, r$ and $k$ can be generated as follows. First three positive integers $u, v, c$ are selected such that $v$ is odd and $c^v\alpha^u$ has approximately the size of the prime to construct. Next the smooth divisors of $c^v\alpha^u - 1$ are computed. This can be done easily using for example trial division or Pollard-rho factorization since only the small prime factors of $c^v\alpha^u - 1$ are wanted. If there exists a smooth divisor $d > c$ of $c^v\alpha^u - 1$ such that $p = c^v\alpha^u - d^v$ is prime, $\frac{p-1}{2} - u$ is relatively prime to $p-1$ and $\alpha$ is a generator of $\mathbb{F}_p^*$ then it is possible to construct a trapdoor as follows.

$$\begin{aligned} r &= c\frac{p-1}{d} \\ k &\equiv v^{-1}\left(\frac{p-1}{2} - u\right) \pmod{p-1} \end{aligned}$$

From $d|(c^v a^u - 1)$ follows that $d$ divides $p-1$. Thus $r$ satisfies the precondition of Theorem 6.1. Because of $\alpha^u c^v \equiv d^v \pmod p$ we have $\alpha^{-u} \equiv (cd^{-1})^v \pmod p$ and $r^v \equiv (-cd^{-1})^v \equiv (-1)\alpha^{-u} \equiv \alpha^{(p-1)/2-u} \pmod p$. Thus $r \equiv \alpha^k \pmod p$ and Theorem 6.1 can be applied.

A heuristic analysis shows that the runtime of this method depends on the size of $c$ and should find a trapdoor after trying $O((\ln p)(\ln\ln p)^2)$ values for $p$. An example for such a trapdoor, which has been generated in less than a day on a SPARC IPC, is given by the values $\alpha = 5, u = 227, v = 1, c = 1629$ and $d = 2^2 \cdot 7 \cdot 23 \cdot 47 \cdot 78541 \cdot 3489781$.

### 6.1.4 Countermeasures

The attacks shown in Section 6.1.2 and 6.1.3 can be avoided if signatures $(r, s)$ are considered to be valid only if — additionally to the other conditions — $r$ is not divisible by a large prime divisor $q$ of $p-1$. This condition should always be checked by the verifier. Moreover, an authorized signer will almost always generate a valid signature since it is very unlikely that he randomly generates an $r$ that is divisible by $q$. Such a condition has been included in the digital signature standard (DSS) [Nat94]. Hence the DSS is not susceptible to the attacks presented in this chapter.

Alternatively trapdoors may be avoided if the authority that is choosing the public parameters $p$ and $\alpha$ is forced to use an algorithm like the one proposed by NIST for the generation of $p$ in DSS. The values produced by this algorithm allow to verify publicly that the parameters have indeed been generated by the algorithm. This would make it very hard for a dishonest authority to create a trapdoor. The two methods to generate trapdoors shown in Section 6.1.3 indicate that both $p$ and $\alpha$ must be generated with this algorithm if no other steps to prevent the attacks are taken.

Yet another possibility to avoid the attacks might be to modify the equations for signature generation and verification (see for example [HMP94a] for an overview of ElGamal variants). Such a variant must be chosen carefully since other problems may arise. For example if a signature on $h$ is computed by $r \equiv \alpha^k \pmod p$ and $s \equiv x_A + hkr \pmod{p-1}$ and therefore verified by $\alpha^s \equiv y_A r^{hr} \pmod p$ then a chosen message attack is possible if the signer can be forced to sign a message $h$ where $\gcd(p-1, h)$ is large. Any such signature leaks information about the secret key $x_A$ since $s \equiv x_A + hkr \pmod{p-1}$ implies $s \equiv x_A \pmod{\gcd(p-1, h)}$. A special case of this attack has been discussed in [HMP94a].

### 6.1.5 An ElGamal scheme over $\mathbb{Z}/(n)$

In [Sar90] Saryazdi has proposed a variant of the ElGamal signature system using $(\mathbb{Z}/(n))^*$ where $n$ is the product of two large primes $p$ and $q$. The author hopes that the security of the proposed signature system relies on the factoring problem and the discrete logarithm problem. Moreover, the existential attack shown in [Gam85] seems not possible in that scheme, because this attack is possible only when the order of the group $(\mathbb{Z}/(n))^*$ is known. However, Horster et al. observed in [HMP94b] that the signatures leak information that can be used to factor the modulus. Even though their attack does not work as described it is possible to modify the attack in such a way that it works for Saryazdi's scheme as well as the improved scheme proposed in [HMP94b]. Moreover, we will show that computing discrete logarithms in only one of the two groups $\mathbb{F}_p^*$ or $\mathbb{F}_q^*$ is sufficient to break the scheme.

**Description of the scheme.** In Saryazdi's variation of the ElGamal signature scheme every user $A$ chooses two large primes $p$ and $q$ and computes $n = pq$. Then he tries to find an integer $\alpha$ with order $\lambda(n)$ in $(\mathbb{Z}/(n))^*$. Furthermore he chooses a random element $x_A$ and computes $y_A = \alpha^{x_A} \pmod n$. $p, q$ and

$x_A$ are kept secret whereas $n, \alpha$ and $y$ are published as $A$'s public key. To sign $h \in \mathbb{Z}/(n)$ user $A$ chooses a random number $k \in (\mathbb{Z}/(n))^*$ and computes

$$\begin{aligned} r &\equiv \alpha^k \pmod{n} && (6.1)\\ s &\equiv (h - x_A r)k^{-1} \pmod{\varphi(n)} && (6.2) \end{aligned}$$

Then $(r, s)$ is the signature on $h$ where $h$ is either the message or the hash value of a message. A verifyer accepts a signature $(r, s)$ on $h$ if $1 \le r < n$ and

$$\alpha^h \equiv (y_A)^r r^s \pmod{n}.$$

This scheme does not allow a modulus $n$ that is common to all users, since every user has to know $\varphi(n)$ in order to be able to compute signatures.

**Description of the attack.** Assume that $t$ is a small prime that divides $p-1$ but not $q-1$. Assume further that $(r_i, s_i) : 1 \le i \le d$ are known signatures on $h_i$ such that $t$ divides $s_i$ and such that the system

$$\begin{aligned} \sum_{i=1}^{d} c_i h_i &\equiv 0 \pmod{t} && (6.3)\\ \text{and} \quad \sum_{i=1}^{d} c_i r_i &\equiv 0 \pmod{t} && (6.4) \end{aligned}$$

has a nontrivial solution (i.e. $c_i \not\equiv 0 \pmod{t}$ for at least one $i$) in the coefficients $c_i$. Such a solution to (6.3) and (6.4) can always be found when at least 3 signatures with $t|s_i$ are known. If we set $B := \sum_{i=1}^{d} c_i h_i$ and $C := \sum_{i=1}^{d} c_i r_i$ then we have

$$\alpha^B \equiv (y_A)^C \prod_{i=1}^{d} r_i^{c_i s_i} \pmod{n}. \qquad (6.5)$$

Since $t$ divides all exponents in (6.5) we can compute the gcd of $n$ and

$$\alpha^{B/t} - (y_A)^{C/t} \prod_{i=1}^{d} r_i^{c_i s_i / t}.$$

This will factor $n$ if exactly one of the following two equations are satisfied.

$$\begin{aligned} \alpha^{B/t} &\equiv (y_A)^{C/t} \prod_{i=1}^{d} r_i^{c_i s_i / t} \pmod{p} && (6.6)\\ \alpha^{B/t} &\equiv (y_A)^{C/t} \prod_{i=1}^{d} r_i^{c_i s_i / t} \pmod{q} && (6.7) \end{aligned}$$

Since $t$ is relatively prime to $q-1$ it follows from (6.5) that (6.7) is satisfied. Since $\alpha$ is a generator modulo $p$ it follows that (6.6) is satisfied if and only if

$$B/t \equiv x_A(C/t) + \sum_{i=1}^{d} k_i(c_i s_i/t) \pmod{p-1}$$

or equivalently

$$B \equiv x_A C + \sum_{i=1}^{d} k_i c_i s_i \pmod{t(p-1)} \tag{6.8}$$

is satisfied. From (6.2) follows only $B \equiv x_A C + \sum_{i=1}^{d} k_i c_i s_i \pmod{\varphi(n)}$. Let $t^z$ be the maximal power of $t$ dividing $\varphi(n)$. It follows that $t^z$ divides $p-1$ and hence $t^{z+1}$ divides $t(p-1)$. Thus we expect that (6.8) is only satisfied with probability about $1/t$.

It should be noted that an attacker does not know which primes $t$ may be successful since he does not know $p$ and $q$ and can generally not determine whether a given $t$ divides $p-1$ but not $q-1$. But this is not a big problem because an attacker can simply try all small primes $t$ that are a divisor of some $s_i$'s.

If the factorization of $n$ can be discovered then signatures can be found if computing discrete logarithms in $\mathbb{F}_p^*$ but not necessarily in $\mathbb{F}_q^*$ is feasible, provided that $p$ is not much smaller than $q$. This can be done as follows.

We may assume that the attacker knows a valid signature $(r, s)$ on $h$. This may be a previously given signature or one that was generated using the existential forgery described in [Gam85]. If he wants to generate a signature on $h'$ then he uses the Chinese Remainder Theorem to compute $0 \le r' < q(q-1)$ such that

$$\begin{aligned} r' &\equiv rh^{-1}h' \pmod{q-1} \\ r' &\equiv r \pmod{q} \end{aligned}$$

If $p$ is not much smaller than $q$ then he will find an $r' < n$ in reasonable time. If we set $s_q \equiv sh^{-1}h' \pmod{q-1}$ then we have

$$\alpha^{h'} \equiv (y_A)^{r'} r'^{s_q} \pmod{q}$$

Assuming that the discrete logarithm in $\mathbb{F}_p^*$ is feasible it is possible to solve

$$h' \equiv \log_\alpha(y_A) r' + \log_\alpha(r') s_p \pmod{p-1}$$

for $s_p$ if $\log_\alpha(r')$ is relatively prime to $p-1$. If additionally the system

$$\begin{aligned} s' &\equiv s_p \pmod{p-1} \\ s' &\equiv s_q \pmod{q-1} \end{aligned}$$

is solvable then $(r', s')$ is a valid signature on $h'$.

**Remarks.** If $t$ divides both $p-1$ and $q-1$ then it follows from (6.2) that

$$B \equiv x_A C + \sum_{i=1}^{d} k_i s_i \pmod{\varphi(n)}$$

is satisfied. This implies that (6.8) and the analogous equation modulo $q-1$ are satisfied. The attack does therefore not work when $t$ divides $p-1$ and $q-1$. The attack described in [HMP94b] considers the case $t=2$ only and does not work therefore. However, it would work when equation (6.2) is reduced modulo $\lambda(n)$ instead of $\varphi(n)$ provided that the maximal powers of 2 in $p-1$ and $q-1$ are not the same The authors of [HMP94b] propose that only signatures $(r,s)$ where $s$ is odd are allowed. This proposal does not prevent the attack described here since $s$ can still be divisible by a small prime divisor $t$.

Another consequence of the note above is that the attack does not work when the smooth parts of $p-1$ and $q-1$ are equal. Thus the attack presented here can be avoided by choosing $p-1$ and $q-1$ such that $(p-1)/2$ and $(q-1)/2$ are prime, since then no small prime $t$ exists dividing $p-1$ but not $q-1$. Some of the extensions proposed in [HMP94b] seem to avoid the attack presented here too. Other variants of ElGamal based on the discrete logarithm problem and the factoring problem have been proposed by Brickell and McCurley [BM92] and Harn [Har94].

## 6.2 A chosen-message attack for Lucas-based RSA

A cryptosystem based on Lucas functions was proposed in [MN81] and analyzed in [MN81] by Müller and Nöbauer. The Lucas functions play the same role as the exponentiation in RSA. Because of equation (2.15), which shows the close connection between exponentiation and Lucas functions, it is not very surprising that it is possible to replace the exponentiation in RSA by Lucas functions.

This scheme can briefly be described as follows. Each user choses two large primes $p$ and $q$ and and exponent $e$ that is relatively prime to $(p^2-1)(q^2-1)$, computes $n = pq$ and publishes $n$ and $e$ as his public key. The corresponding $d \equiv e^{-1} \pmod{(p^2-1)(q^2-1)}$ is kept secret. A message $m$ is encrypted by computing $y \equiv V_e(m,1) \pmod{n}$. Decryption is done by $m \equiv V_d(y,1) \pmod{n}$.

It has been claimed in [Smi93] that no chosen message attack for Lucas-based RSA is possible. A chosen-message forgery that needs two chosen messages has been described in [BBL95]. Another attack that needs only one chosen message is described here.

**Chosen-message attack for Lucas-based RSA**

Input: *A message $m$, the public key $n, e$ of a user*

Step 1: *Choose a random integer $a$ relatively prime to $e$.*

Step 2: *Compute $x \equiv V_a(m,1) \pmod{n}$.*

Step 3: *Get the signature* $y \equiv V_d(x,1) \pmod{n}$ *from the user.*
Step 4: *Use Euclid's algorithm to find* $s, t$ *such that* $as - et = 1$.
Step 5: *Compute*

$$\begin{aligned} V_{sad}(m,1) &\equiv V_s(y,1) \pmod{n} && (6.9)\\ U_{sad}(m,1) &\equiv \frac{U_a(m,1)U_s(y,1)}{U_e(y,1)} \pmod{n} && (6.10)\\ V_d(m,1) &\equiv \frac{V_{sad}(m,1)V_t(m,1)}{2} \\ &\quad - \frac{\Delta U_{sad}(m,1)U_t(m,1)}{2} \pmod{n} && (6.11) \end{aligned}$$

*Proof:* Equation (6.9) follows from (2.27) since

$$V_s(y,1) \equiv V_s(V_{ad}(m,1),1) \equiv V_{sad}(m,1) \pmod{n}.$$

Using (2.28) we have

$$\begin{aligned} U_{sad}(m,1)U_e(y,1) &\equiv U_s(V_{ad}(m,1))U_{ad}(m,1)U_e(V_{ad}(m,1),1) \\ &\equiv U_s(V_{ad}(m,1))U_{ade}(m,1) \\ &\equiv U_s(V_{ad}(m,1))U_a(m,1) \pmod{n} \end{aligned}$$

This shows that Equation (6.10) is satisfied. It follows from $as - et = 1$ that $V_d(m,1) = V_{sad-det}(m,1) = V_{sad-t}(m,1)$. Hence using (2.29) and (2.24) we get Equation (6.11). □

*Remark.* This attack is easier to understand in the ring $\mathcal{O}_{\sqrt{\Delta}}$ with $\Delta = m^2 - 4$. It follows from

$$U_a(m,1) \equiv U_{ade}(m,1) \equiv U_{ad}(m,1)U_e(V_{ad}(m,1),1) \pmod{n}$$

that $U_{ad}(m,1)$ is computable from $m$ and $V_{ad}(m,1)$. If $\alpha = m/2 + \sqrt{\Delta}/2$ then the signature $V_{ad}(m,1)$ on the message $V_a(m,1)$ can be used to compute

$$\alpha^{ad} \equiv V_{ad}(m,1)/2 + U_{ad}(m,1)\sqrt{\Delta}/2.$$

Once $\alpha^{ad}$ is known it is possible to derive $\alpha^d = V_d(m,1)/2 + U_d(m,1)\sqrt{\Delta}/2$ by

$$\alpha^d \equiv \alpha^{(as-et)d} \equiv \left(\alpha^{ad}\right)^s \alpha^{-t} \pmod{n}.$$

# Bibliography


[AGP94a] W. R. Alford, A. Granville, and C. Pomerance. On the difficulty of finding reliable witnesses. In *Algorithmic number theory*, volume 877 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 1994.

[AGP94b] W. R. Alford, A. Granville, and C. Pomerance. There are infinitely many Carmichael numbers. *Annals of Mathematics*, 140(3):703–722, 1994.

[AH92] L. Adleman and M. Huang. *Primality testing and two dimensional Abelian varieties over finite fields*, volume 1512 of *Lecture Notes in Mathematics*. Springer-Verlag, 1992.

[AM94] L. M. Adleman and K. S. McCurley. Open problems in number theoretic complexity, II. In L. M. Adleman and M.-D. Huang, editors, *Algorithmic Number Theory*, volume 877 of *Lecture Notes in Computer Science*, pages 291–322. Springer-Verlag, 1994.

[Arn91] F. Arnault. Le test de primalité de Rabin-Miller: Un nombre composé qui le passe. Technical report, Université de Poitiers Département de Mathématiques, November 1991.

[Arn93] F. Arnault. Carmichaels fortement pseudo-premiers, pseudo-premiers de Lucas. Technical report, Université de Poitiers Département de Mathématiques, January 1993.

[Arn95] F. Arnault. Rabin-Miller primality test: Composite numbers which pass it. *Mathematics of Computation*, 64(209):355–361, January 1995.

[Bac85] E. Bach. Analytic methods in the analysis and design of number-theoretic algorithms. Technical report, M.I.T. Press, Cambridge, 1985.

[BBL95] D. Bleichenbacher, W. Bosma, and A. Lenstra. Some remarks on Lucas-based cryptosystems. In D. Coppersmith, editor, *Advances* *in Cryptology – CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 386–396. Springer-Verlag, 1995.

[BC90] J. Bos and M. Coster. Addition chain heuristics. In G. Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435, pages 400–407. Springer-Verlag, 1990. Lecture Notes in Computer Science.

[BFSe92] T. Beth, M. Frisch, and G. Simmons (eds). *Public-key Cryptography, State of the Art and Future Directions*, volume 578 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1992.

[BGMW93] E. F. Brickell, D. M. Gordon, K. S. McCurley, and D. B. Wilson. Fast exponentiation with precomputation. In R. A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 200–207, 1993.

[BM92] E. F. Brickell and K. S. McCurley. An interactive identification scheme based on discrete logarithms and factoring. *Journal of Cryptology*, 5(1):29–39, 1992.

[Bre89] D. M. Bressoud. *Factorization and Primality Testing*. Springer-Verlag, 1989.

[BvdH90] W. Bosma and M.-P. van der Hulst. *Primality Proving with Cyclotomy*. PhD thesis, University of Amsterdam, 1990.

[BW80] R. Baillie and S. Wagstaff. Lucas pseudoprimes. *Mathematics of Computation*, 35(152):1391–1417, October 1980.

[Car12] R. D. Carmichael. On composite numbers $P$ which satisfy the Fermat congruence $a^{P-1} \equiv 1 \bmod P$. *American Math. Monthly,*, 19:22–27, 1912.

[CL87] H. Cohen and A. K. Lenstra. Implementation of a new primality test. *Mathematics of Computation*, 48(177):103–121, January 1987.

[Coh93] H. Cohen. *A Course in Computational Algebraic Number Theory*. Number 138 in Graduate Texts in Mathematics. Springer-Verlag, 1993.

[Dav92] J. H. Davenport. Primality testing revised. *Proc. ISSAC 92, ACM Press, New York*, 1992.

[Dia82] H. G. Diamond. Elementary methods in the study of the distribution of prime numbers. *Bulletin of the American Mathematical Society (New Series)*, 7(3):553–589, January 1982.

[Dic97] L. E. Dickson. The analytic representation of substitutions on a prime number of letters with a discussion of the linear group. *Annals of Math.*, 11:65–120,161–183, 1896/97.

[DLS81] P. Downey, B. Leong, and R. Sethi. Computing sequences with addition chains. *SIAM Journal on Computing*, 10(3):638–646, August 1981.

[Fla91] A. Flammenkamp. Math. Diplomarbeit. Univ. Bielefeld, 1991.

[Gam85] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. C. Chaum, editors, *Advances in Cryptology: Proceedings of CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag, 1985.

[Gor92] D. M. Gordon. Designing and detecting trapdoors for discrete log cryptosystems. In E. F. Brickell, editor, *Advances in Cryptology–CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 66–75. Springer-Verlag, 1992.

[Gor93] D. M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM J. Disc. Math.*, 6(1):124–138, February 1993.

[Har94] L. Harn. Public-key cryptosystem design based on factoring and discrete logarithm. *IEE Proc. Comput. Digit. Tech.*, 141(3):193–195, 1994.

[HMP94a] P. Horster, M. Michels, and H. Petersen. Generalized ElGamal signatures for one message block. Technical Report TR-94-3, University of Technology Chemnitz-Zwickau, May 1994.

[HMP94b] P. Horster, M. Michels, and H. Petersen. Meta-ElGamal signature schemes using a composite module. Technical Report TR-94-16-E, University of Technology Chemnitz-Zwickau, November 1994.

[HW79] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers.* Clarendon Press, Oxford, 5th edition, 1979.

[IR90] K. Ireland and M. Rosen. *A Classical Introduction to Modern Number Theory.* Graduate Texts in Mathematics. Springer-Verlag, 2nd edition, 1990.

[Jae93] G. Jaeschke. On strong pseudoprimes to several bases. *Mathematics of Computation*, 61(204):915–926, 1993.

[Knu81] D. E. Knuth. *The art of computer programming, Seminumerical Algorithms*, volume 2. Addison Wesley, 2nd edition, 1981.

[Leh81] D. Lehmer. On Fermat's quotient, base two. *Mathematics of Computation*, 36:289–290, 1981.

[Len84] H. Lenstra. Divisors in residue classes. *Mathematics of Computation*, 42:331–340, 1984.

[Len90] A. K. Lenstra. Primality testing. In *Proc. of the AMS Symposia in Applied Mathematics: Computational Number Theory and Cryptography*, pages 13–25. American Mathematical Society, 1990.

[LM83] R. Lidl and W. B. Müller. Permutation polynomials in RSA-cryptosystems. In D. Chaum, editor, *Proceedings of CRYPTO '83*, New York, 1983. Plenum.

[LM91] R. Lidl and W. B. Müller. Generalizations of the Fibonacci pseudoprimes test. *Discrete Mathematics*, 92:211–220, 1991.

[LMT93] R. Lidl, G. L. Mullen, and G. Turnwald. *Dickson Polynomials.* Longman Scientific & Technical (copublished with John Wiley & Sons, Inc.), 1993.

[Mal77] D. E. G. Malm. On Monte-Carlo primality tests. *Notices Amer. Math. Soc.*, pages A–529, 1977.

[Mau95] U. M. Maurer. Fast generation of prime numbers and secure public-key cryptographic parameters. *Journal of Cryptology*, 8(3):123–155, 1995.

[MN81] W. B. Müller and W. Nöbauer. Some remarks on public-key cryptosystems. *Studia Sci. Math. Hungar.*, 16:71–76, 1981.

[Mon] P. L. Montgomery. Evaluation recurrences of form $X_{m+n} = f(X_m, X_n, X_{m-n})$ via Lucas chains. to appear in Fibonacci Quarterly.

[Mon80] L. Monier. Evaluation and comparison of two efficient probabilistic primality testing algorithms. *Theoretical computer science*, 12:97–108, 1980.

[Mon87] P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, January 1987.

[Mor90] F. Morain. Distributed primality proving and the primality of $(2^{3539}+1)/3$. In *Advances in Cryptology - EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 110–123. Springer-Verlag, 1990.

[Mor92] F. Morain. Prime values of partition numbers and the primality of $p(1840926)$. Technical Report LIX/92/RR/11, Laboratoire d'Informatique de l'Ecole Polytechnique (LIX), F-91128 Palaiseau Cedex, FRANCE,, 1992.

[MQV95] A. Menezes, M. Qu, and S. Vanstone. Key agreement and the need for authentication. PKS, November 1995.

[Nat94] National Institute of Standards and Technology (NIST). *FIPS Publication 186: Digital Signature Standard*, May 19, 1994.

[New80] D. J. Newman. Simple analytic proof of the prime number theorem. *American Math. Monthly*, 87:693–696, 1980.

[Nil82] N. J. Nilsson. *Principles of Artificial Intelligence*. Springer-Verlag, 2nd edition, 1982.

[Nöb85] R. Nöbauer. Cryptanalysis of the Rédei-scheme. *Contributions to General Algebra 3, Proceedings of the Vienna Conference*, pages 255–264, 1985.

[PF88] D. Porto and Filipponi. A probabilistic primality test based on the properties of certain generalized Lucas numbers. In *Advances in Cryptology – EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 211–223, 1988.

[PH78] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Trans. Inform. Theory*, IT-24:106–110, January 1978.

[Pin93a] R. Pinch. The Carmichael numbers up to $10^{15}$. *Mathematics of Computation*, 61(203):381–391, July 1993.

[Pin93b] R. Pinch. The Carmichael numbers up to $10^{16}$. Preprint, 1993.

[Pin93c] R. Pinch. The pseudoprimes up to $10^{13}$. Preprint, 1993.

[Pin93d] R. G. E. Pinch. Some primality testing algorithms. *Notices of the American Mathematical Society*, 40, November 1993.

[Poc16] H. C. Pocklington. The determination of the prime or composite nature of large numbers by Fermat's theorem. *Proc. Cambridge Philos. Soc.*, 18:29–30, 1914-1916.

[PSW80] C. Pomerance, J. L. Selfridge, and S. S. Wagstaff. The pseudoprimes to $25 \cdot 10^9$. *Mathematics of Computation*, 35(151):1003–1026, July 1980.

[Réd48] L. Rédei. Über eindeutig umkehrbare Polynome in endlichen Körpern. *Acta Sci. Math.*, 11:71–76, 1946–48.

[Rie85] H. Riesel. *Prime Numbers and Computer Methods for Factorization.* Birkhäuser, 1985.

[RLS+93] R. A. Rueppel, A. K. Lenstra, M. E. Smid, K. S. McCurley, Y. Desmedt, A. Odlyzko, and P. Landrock. Panel discussion: Trapdoor primes and moduli. In R. A. Rueppel, editor, *Advances in Cryptology — EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 194–199. Springer-Verlag, 1993.

[Sar90] S. Saryazdi. An extension to ElGamal public key cryptosystem with a new signature scheme. In E. Ankan, editor, *Communication, Control, and Signal Processing*, pages 195–198. Elsevier, 1990.

[Sch73] I. Schur. Arithmetisches über die Tschebyscheffsen Polynome. In *Ges. Abhandlungen*, volume 3, pages 422–453. Springer-Verlag, Berlin-New York, 1973.

[SD93] J. Sauerbrey and A. Dietel. Resource requirements for the application of addition chains in modulo exponentiation. In R. A. Rueppel, editor, *Advances in Cryptology — EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 174–182. Springer-Verlag, 1993.

[Smi93] P. Smith. LUC public-key encryption. *Dr. Dobb's Journal*, pages 44–49, January 1993.

[TC87] Y. Tsai and Y. Chin. A study of some addition chain problems. *International Journal of Computer Mathematics*, 22:117–134, 1987.

[Wil85] H. C. Williams. Some public-key crypto-functions as intractable as factorizations. *Cryptologia*, 9(3):223–237, July 1985.

[Yac90] Y. Yacobi. Exponentiating faster with addition chains. In *Advances in Cryptology: – EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 222–229. Springer-Verlag, 1990.

# Index

# Lebenslauf

Am 4. August 1964 wurde ich in Langenthal geboren. Dort besuchte ich von 1970 bis 1984 die Primar- und Sekundarschule, Untergymnasium und Gymnasium und schloss mit der Matura Typus C ab. Anschliessend begann ich ein Informatikstudium an der Universität Bern, welches ich im April 1991 mit dem Diplom abschloss. In der Zeit von 1988 bis Ende 1991 arbeitete ich auch bei der Firma Badertscher Informatik als Programmierer von Datenbankanwendungen. Im Januar 1992 trat ich eine Assistentenstelle bei Prof. U. Maurer an.