Sapienza
Università di Roma

Integer Sequences in Cryptography: A New Generalized Family and its Application

Dipartimento di Informatica
Dottorato di Ricerca in Informatica - XXXVII Ciclo

Mario Raso
ID number 687516

Thesis Advisor
Prof. Daniele Venturi

Academic Year 2024/2025

Integer Sequences in Cryptography: A New Generalized Family and its Application

Tesi sperimentale retrospettiva monocentrica. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: raso@di.uniroma1.it

To my family

Abstract

Integer sequences play a pivotal role in cryptography, acting as foundational elements for numerous cryptographic algorithms. This comprehensive investigation examines integer sequences that have significantly impacted the sector in domains such as key generation, hash function design, and encryption protocol development, including their specific implementations. We conduct an unprecedented systematic review of existing literature, analysing fundamental properties of these sequences and detailing their contributions to well-established cryptographic areas. In addition, the research emphasises the various strengths and limitations associated with these sequences, as well as their practical applications in the realm of digital information security. This is accomplished by developing a categorisation framework that facilitates mapping of their contributions. Furthermore, this framework can be used as a reference point for future analyses in this field.

As a result, our initial emphasis was on the thorough investigation and conceptualisation of a novel integer sequence generation model. This model holds the potential to be remarkably unique, capable of unifying and encompassing both existing integer sequences and those yet to be discovered into a cohesive and comprehensive framework. Following this foundational work, our attention shifted to analysing the prospective applications of this model, particularly within the field of cryptography. Here, we honed in on the intricate concept of randomness, delving into a detailed analysis of its potential significance and the various implications it may hold for advancing cryptographic techniques and security protocols.

Randomness is a key ingredient in every area of cryptography and producing it should not be left to chance. Unfortunately, it is very difficult to produce true randomness, and consuming applications often call for large, high quality amounts on boot or in quick succession. To meet this requirement, we make use of Pseudo-Random Number Generators (PRNGs) which we initialise with a small amount of randomness to produce what we hope to be high quality pseudo-random output.

Therefore, our model has been instrumental in the design of a PRNG which has exhibited statistically significant capabilities, as well as satisfactory performance metrics, to corroborate its inherent randomness properties.

In summary, this study emphasises the considerable potential that exists for the ongoing exploration and development of novel applications involving integer sequences within the domain of cryptography. The findings suggest that these sequences could play a pivotal role in enhancing and advancing cryptographic methodologies, opening avenues for new innovations and improvements.

Acknowledgments

I would like to express my deepest gratitude to Sapienza University of Rome for giving me the opportunity to embark on this PhD journey. This academic path has been both intense and stimulating, and it would not have been possible without the support of the institution that welcomed and guided me throughout these years.

A special thank you goes to my advisor, prof. Daniele Venturi, for their invaluable guidance, availability, and intellectual encouragement, which have greatly enriched my work. Their expertise and support have been fundamental in achieving this milestone.

I am also grateful to all the professors and colleagues from the Department of Computer Science for their support, valuable advice, and insightful discussions that have contributed to my academic and personal growth. A special mention to the administrative staff, whose help has been essential in navigating the bureaucratic challenges of the PhD program.

I sincerely thank Sapienza University of Rome for providing me with a stimulating environment, high-quality resources, and the opportunity to engage with esteemed scholars. This PhD has been much more than an academic journey; it has been a life-changing experience that I will always cherish.

Finally, to all those who have been part of this journey, in small and great ways, your support has made all the difference. Thank you for being part of my path and for helping me reach this milestone.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Throughout the history of cryptography, there have been instances of encountering specific sequences of numbers with distinctive characteristics. The most well-known sequence is undoubtedly the set of prime numbers, but other sequences have also been significant in various applications.

As mathematicians investigate integer sequences further, they discover new patterns and connections that are relevant in fields like cryptography. These unique sequences serve as the foundation for cryptographic algorithms and data encryption, presenting opportunities to develop secure communication systems and protect sensitive information based on their intricate relationships. Therefore, the study of integer sequences is not only a fascinating pursuit in pure mathematics but also a driving force behind innovative advancements in technology and science. Its impact, from providing fundamental insights into number theory to enabling secure communications and efficient data processing, underscores the far-reaching significance of integer sequences in our modern world.

Therefore, studying new models to generate families of integer sequences is useful for purposes aimed at by cryptography, particularly for those applications that require randomness.

The absence of guarantees regarding the randomness of the generated numbers can result in significant vulnerabilities within cryptographic protocols, which can be exploited by attackers. A notable instance is the issue within the one version of the Debian Linux distribution [CVE08], where a commented section in the OpenSSL PRNG code contributed to inadequate entropy collection, consequently exposing the TLS and SSH protocols to tangible attacks. In work [LHA$^+$12], it was shown that a significant fraction of RSA keys have common prime factors. In another contribution [HDWH12], an analysis was provided that reveals how Linux PRNG leads to the production of low-entropy keys, particularly when generated during boot time. Furthermore, cryptographic algorithms are particularly susceptible to flaws in the foundational random number generation process. For example, numerous studies have shown that using a weak pseudo-random number generator to create nonces for the DSS signature algorithm can lead to rapid recovery of the secret key after observing only a few signatures (refer to [NS02] and related citations).

This highlights the critical necessity for a thorough assessment of PRNGs under well-defined security standards.

1.2 Our Contributions

Based on a thorough exploration of the bibliographic references outlined in the preceding paragraph, it becomes apparent that there is a significant lack of systematic review within the existing literature that thoroughly describes the use of integer sequences in cryptography. Therefore, our research began by filling this gap within the targeted scientific domain. In doing so, we intend to establish a foundational framework that will facilitate the systematic examination of the use of integer sequences in cryptographic practices.
The contributions presented within the context of this comprehensive systematic literature review are as follows:

1. By synthesising the findings, we aim to provide a comprehensive analysis of existing research. This seeks to offer a clear, detailed, and organised overview of current trends, research gaps, and emerging directions within the field. The comprehensive analysis helps identify areas that require further exploration and emerging topics that are gaining attention.

2. We offer critical insights and form recommendations for practitioners and researchers based on the current state of the field. This review serves as a valuable resource for both researchers and practitioners, guiding their future efforts. In addition, it fosters a deeper understanding of the cryptographic landscape and its complexities, helping to advance both theoretical and applied research.

3. We propose a conceptual framework for future research. This framework outlines potential methodologies and theoretical perspectives that could be employed to address the limitations and gaps identified in previous studies. By suggesting new approaches and theoretical underpinnings, we aim to stimulate innovative research directions and contribute to the advancement of the field.

Subsequently, on the basis of the analysis of the literature, we achieved the following research results:

Result 1: Definition of a sequence generation model that is largely novel and capable of aggregating existing sequences into a single family.

Result 2: Application of the aforementioned sequence generation model for the development of a PRNG on which statistics have been performed to validate its randomness.

1.3 Thesis organisation

The primary structure of the thesis is described as follows:

- **Chapter 2** - We first recall some basic notions about sequences of integers and the main tools for analysing their behaviour.
- **Chapter 3** - Outlines the primary sequences that have been employed in the field of cryptography throughout history. In particular, the classification of the uses of each sequence will be thoroughly and systematically carried out, with consideration given to the macro-areas below, which will provide the overarching framework for this process of categorisation and arrangement:
  - *Foundations.* The paradigms, approaches, and techniques used to conceptualise, define, and provide solutions to natural cryptographic problems.
  - *Cryptographic hash functions.* Algorithm that computes a numerical value (called a hash value) on a data file or electronic message that is used to represent that file or message and depends on the entire contents of the file or message. A hash function can be considered to be a fingerprint of the file or message.
  - *Secret-key cryptography.* Also referred to as symmetric key cryptography, is a cryptographic algorithm that uses the same secret key for its operation and, if applicable, for reversing the effects of the operation (e.g., an AES key for encryption and decryption).
  - *Public-key cryptography.* Alternatively, named asymmetric cryptography is a set of three cryptographic algorithms (KeyGen, Encrypt, and Decrypt) that can be used by two parties to send secret data over a public channel. Also known as an asymmetric encryption scheme.
  - *Cryptographic protocols.* These protocols consist of a collection of rules and processes in which cryptographic algorithms are used to guarantee secure communication and data exchange.
  - *Implementation.* Application that performs a cryptographic function.
  - *Attacks and cryptanalysis.* The study of techniques to attempt to defeat cryptographic techniques and information system security. This includes the process of looking for errors or weaknesses in the implementation of an algorithm or of the algorithm itself.
  - *Steganography.* The art, science and practice of communicating in a way that hides the existence of the communication, embedding data within other data to conceal it.

  Furthermore, in the corresponding subsections titled "Further details", for each sequence, one of the applications in cryptography described synthetically is chosen and analysed in depth. In the concluding segment of Chapter, in Section 3.2 we provide a summary analysis highlighting the macro-areas of cryptography in which the sequences have provided a significant contribution.
- **Chapter 4** - We describe the construction of a new model for generating integer sequences capable of defining a new family of integer sequences.
- **Chapter 5** - The previously discussed model is utilised in cryptographic applications through the creation of a PRNG (Pseudo-Random Number Generator), followed by a statistical evaluation of its randomness.

Chapter 2

Preliminaries and Notation

2.1 Integer sequences

Integer sequences are found in various areas of mathematics and have practical uses in fields such as computer science, physics, and engineering. They are frequently used to represent real-world problems and phenomena, and understanding their characteristics can provide valuable insights and solutions.

An essential part of comprehending integer sequences is to identify patterns and links between the terms. This includes examining how the sequence behaves, recognising recurring relationships, and investigating any fundamental mathematical principles that govern its development. Furthermore, researching integer sequences often involves discovering connections to other branches of mathematics such as number theory, combinatorics, and algebra. By exploring these connections, mathematicians can gain a deeper understanding of the nature of the sequences' broader significance.

2.2 On-Line Encyclopedia of Integer Sequences (OEIS)

The *On-line Encyclopedia of Integer Sequences* (OEIS) is a comprehensive and widely used reference database that catalogues integer sequences [oei]. In 1964, Neil Sloan began collecting integer sequences as a graduate student to support his research in combinatorics. The database was initially stored on a punch card. He published two book selections from the database:

– *A Handbook of Integer Sequences* (1973) [Slo73], which contains 2372 sequences in lexical order and assigns numbers from 1 to 2372.

– *The Encyclopedia of Integer Sequences* (1995)[SP95], which contains 5488 sequences and assigns numbers M from M0000 to M5487. The encyclopedia includes references to the corresponding sequence in the *A Handbook of Integer Sequences* (in the few initial terms that may vary), i.e., N of N0001 to N2372 (instead of 1 to 2372). The encyclopedia contains the *A* numbers used in OEIS, while the manual does not.

These books were highly appreciated, and especially after the second edition, mathematicians provided a steady flow of new sequences for Sloane. The collection became unmanageable in book form and, when the database reached 16000 entries, Sloan decided to go online, initially as an e-mail service in August 1994 and shortly thereafter as a website in 1996. As a spin-off of database work, Sloane founded the *Journal of Integer Sequences* in 1998. The database continues to grow at an annual rate of about 10000 entries. Sloane has been personally managing his sequences for nearly 40 years, but since 2002 a team of associate editors and volunteers has helped maintain the database.

In 2004, Sloane celebrated the addition of the 100000th sequence to the database, and in 2006 the user interface was overhauled and more advanced search capabilities were added. Later, in 2009, he transferred OEIS intellectual property and hosting to the OEIS Foundation. Sloane is the chairman of the OEIS Foundation and in 2010 an OEIS wiki at OEIS.org was created to simplify the collaboration of the OEIS editors and contributors. The OEIS registers information on integer sequences of interest to both professional and amateur mathematicians and is widely cited. As of April 2023, it contained more than 360000 sequences and is the largest database of its kind. To each sequence OEIS assigns a unique identification code of the AXXXXXX type with which a new database entry is established. Each entry contains key terms such as sequences, keywords, mathematical motives, literature links, and other options such as graphic generation and musical representations of sequences. The database can be searched by keywords, subsequences, or one of 16 fields. It serves as a valuable resource for mathematicians, researchers, and enthusiasts who study and analyse the patterns and properties of integer sequences, providing insights into mathematical patterns, and serving as a reference for researchers exploring the vast field about them.

The OEIS promotes a strong sense of unity and cooperation within the field of mathematics. By offering a platform for individuals to share their expertise, the encyclopedia continues to develop and grow, benefiting from the collective knowledge of mathematicians, researchers, educators, and students around the world.

Community involvement in the encyclopedia goes beyond simply submitting new sequences and corrections. Users can participate in discussions, exchange ideas, and provide suggestions for improving existing sequences. This collaborative approach not only improves the quality and precision of the database, but also builds a vibrant community where individuals can learn from one another while collectively advancing their understanding of integer sequences.

In addition, the encyclopedia fosters collaboration by highlighting how different sequences are connected and encouraging users to explore the relationships among them. By recognising patterns, similarities, and dependencies between various sequences, users contribute to expanding mathematical knowledge while uncovering new connections that enhance research on integer sequences.The interactive nature of OEIS underscores its vitality as a continually developing resource which is shaped by active participation members. As users continue sharing their expertise discoveries, user-generated content remains an integral aspect maintaining position at forefront research valuable insights fostering spirit collaboration among maths specialists enthusiasts.

2.3 Notation and Mathematical Preliminaries

The chapter serves a vital role in establishing a strong foundation for the mathematical context of the work. Its objective is to create a shared and precise language, necessary for an accurate comprehension of the material presented throughout the text. By introducing specific notation and mathematical fundamentals, we aim to establish a solid basis for developing subsequent arguments and proofs, enabling clear communication of complex concepts that will be explored in this chapter as well as throughout the entire work. Consistent notation and precise terminology demonstrate our dedication to accurately treating the material, ultimately contributing to a deeper understanding of the mathematical aspects covered in this work.

Sequence

Definition 2.3.1. *An integer sequence is a sequence (i.e., an ordered list) of integers.*

Remark 1. *A sequence is an enumerated collection of objects in which repetitions are allowed and order matters. Like a set, it contains members (also called elements or terms). The sequence length is determined by the number of elements (possibly infinite).*

Symbolically $(a_n)_{n\in\mathbb{N}}$ denotes a sequence whose nth element is given by the variable a_n. For example:

$$\begin{aligned} a_1 &= 1 \text{ st element of } (a_n)_{n\in\mathbb{N}} \\ a_2 &= 2 \text{ nd element} \\ a_3 &= 3 \text{ rd element} \\ &\vdots \\ a_{n-1} &= (n-1) \text{ th element} \\ a_n &= n\text{th element} \\ a_{n+1} &= (n+1) \text{ th element} \\ &\vdots \end{aligned} \tag{2.1}$$

For the above, we can define a sequence of integers as:

– Explicit formula for its nth term. For example, the sequence $0, 3, 8, 15, \ldots$ is formed according to the formula $n^2 - 1$ for the nth term.

– Relationship between its terms. For example, the sequence $0, 1, 1, 2, 3, 5, 8, 13, \ldots$ (Fibonacci sequence - OEIS: A000045) is formed by starting with 0 and 1 and then adding any two consecutive terms to obtain the next one.

– Alternatively, an integer sequence may be defined by a property that members of the sequence possess and that other integers do not possess. For example, we can determine whether a given integer is a perfect number, even though we do not have a formula for the nth perfect number.

Subsequences. Subsequences are sequences formed from the given sequence by deleting some of the elements without disturbing the relative positions of the remaining elements. For example, the sequence of even positive integers $(2, 4, 6, \ldots)$ is a subsequence of positive integers $(1, 2, 3, \ldots)$. In the case of deletion of some elements, the positions of those elements change. However, the relative positions are preserved.

In formal terms, a subsequence of the sequence $(a_n)_{n\in\mathbb{N}}$ is any sequence of the form $(a_{n_k})_{k\in\mathbb{N}}$, where $(n_k)_{k\in\mathbb{N}}$ is a strictly increasing sequence of positive integers.

Recurrence relation

Definition 2.3.2. *Let a_n be a sequence of numbers. A recurrence relation on a_n is an equation according to which the nth term of a sequence of numbers is equal to some combination of the previous terms.*

More precisely, in the case where only the immediately preceding element is involved, a recurrence relation has the form:

$$a_n = f(n, a_{n-1}) \quad \text{for } n > 0 \tag{2.2}$$

where

$$f : \mathbb{N} \times X \to X \tag{2.3}$$

is a function, where X is a set to which the elements of a sequence must belong. For any $n_0 \in X$, this defines a unique sequence with n_0 as its first element, called the initial value. It is easy to modify the definition to get sequences starting from the term of index 1 or higher. This defines the first-order recurrence relation. A recurrence relation of order k has the form:

$$a_n = f(n, a_{n-1}, a_{n-2}, \ldots, a_{n-k}) \quad \text{for } n \geq k \tag{2.4}$$

where $f : \mathbb{N} \times X^k \to X$ is a function that involves k consecutive elements of the sequence. In this case, the initial values k are needed to define a sequence.
Example. Some examples to explain the order of a relation:

- The Fibonacci sequence (OEIS: A000045) is defined by the recurrence relation $F_n = F_{n-2} + F_{n-1}, n \geq 2$, with the initial conditions $F_0 = 1$ and $F_1 = 1$. The recurrence relation is called a second-order relation because F_n depends on the two previous terms of F.

- The relation $T_n = 2T_{n-1}^2 - kT_{n-3}$ is a third-order recurrence relation. If the values of T_0, T_1 and T_2 are specified, then T_n is fully defined.

- The recurrence relation $S_n = S_{\lfloor n/2 \rfloor} + 5, n > 0$, with $S_0 = 0$ is infinitely ordered. To determine S_n when n is even, you must go back to the terms of $n/2$. Since $n/2$ grows unbounded with n, no finite order can be given to S.

Recurrence relations appear in a natural way when studying several different kinds of problem, like computing increments or decrements of populations with given reproduction rules, colouring pictures with just two colours, computing the number of moves in different games, computing compounded interests, solving geometrical problems, etc.

Linear recurrence

Definition 2.3.3. *Integer sequences that can be described as homogeneous linear recurrence relations with constant rational coefficients.*

Suppose that we have a function $f : \mathbb{N} \to \mathbb{R}$. Setting $a_n = f(n)$ for all $n \in \mathbb{N}$, we term the set $\{a_n\}_{n=1}^{\infty}$ a sequence. Assuming that we know $a_1, \ldots, a_k$ and for $a_n = f(a_{n-1}, \ldots, a_{n-k})$ for some function $f : \mathbb{R}^k \to \mathbb{R}$, we say that $\{a_n\}_{n=1}^{\infty}$ is a recursively defined sequence given by the recurrence relation $a_n = f(a_{n-1}, \ldots, a_{n-k})$.

We say a recurrence relation is linear if f is a linear function or in other words:

$$a_n = f(a_{n-1}, \ldots, a_{n-k}) = s_1 a_{n-1} + \cdots + s_k a_{n-k} + f(n) \tag{2.5}$$

where $s_i, f(n)$ are real numbers [EVDPS+03]. Moreover, a linear recurrence relation is homogeneous if $f(n) = 0$ and non-homogeneous if $f(n) \neq 0$.

Definition 2.3.4. *Homogeneous linear recurrence relations with constant coefficients. An order k homogeneous linear recurrence relation with constant coefficients is an equation of the form:*

$$\sum_{i=0}^{k} c_i a(n-i) = c_0 a(n) + c_1 a(n-1) + c_2 a(n-2) + \cdots + c_k a(n-k) = 0 \tag{2.6}$$

where the k coefficients $c_i (\forall i)$ are constants.

Example. Homogeneous linear recurrences (of order 1) with constant coefficients:

$$\begin{cases} a_0 = 1 \\ a_n = 2a_{n-1}, \quad n \geq 1 \end{cases} \tag{2.7}$$

(OEIS: A000079) Powers of 2 : $a(n) = 2^n$:

$$\{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, \ldots\}. \tag{2.8}$$

From the generating function of powers of 2 (where in the second version the denominator has the form of the recurrence):

$$G_{\{2^n, n \geq 0\}}(x) = \frac{1}{1-2x} = \frac{(x^{-1})^1}{(x^{-1})^1 - 2(x^{-1})^0}, \tag{2.9}$$

and setting x^{-1} to 10^k, we get the form:

$$\frac{\left(10^k\right)^1}{(10^k)^1 - 2(10^k)^0} = \frac{10^k}{10^k - 2} = \sum_{n=0}^{\infty} \frac{2^n}{(10^k)^n}, \quad k \geq 1. \tag{2.10}$$

For example, for the first few values of k, we have (note that overlapping would occur if powers of 1 had more than k digits):

$$\begin{aligned} &k = 1 : 10/8 = 1.25 \left(\text{here } \textstyle\sum_{n=3}^{\infty} \frac{2^n}{(10^n)^n} = 0.01, \text{ and } 1 + 2/10 + 4/100 = (100+20+4)/100 \right) \\ &k = 2 : 100/98 = 1.0204081632653061224489795918\ldots \\ &k = 3 : 1000/998 = 1.0020040080160320641282565130\ldots \\ &k = 4 : 10000/9998 = 1.0002000400080016003200640128\ldots \end{aligned} \tag{2.11}$$

A variant of the above is:

$$\frac{1}{\left(10^k\right)^1 - 2\left(10^k\right)^0} = \frac{1}{10^k - 2} = \sum_{n=0}^{\infty} \frac{2^n}{\left(10^k\right)^{n+1}}, \quad k \geq 1. \tag{2.12}$$

For example, for the first few values of k, we have (note that overlapping occurs when powers of 2 have more than k digits):

$$\begin{aligned}
&k = 1 : 1/8 = 0.125 \left(\text{here } \textstyle\sum_{n=3}^{\infty} \frac{2^n}{\left(10^k\right)^{n+1}} = 0.001, \text{ and } 1/10 + 2/100 + 4/1000 = (100+20+4)/1000 \right) \\
&k = 2 : 1/98 = 0.010204081632653061224489795918\ldots \ (\text{A021102}) \\
&k = 3 : 1/998 = 0.001002004008016032064128256513\ldots \ (\text{A022002}) \\
&k = 4 : 1/9998 = 0.000100020004000800160032006401\ldots
\end{aligned} \tag{2.13}$$

Example. Homogeneous linear recurrences (of order 2) with constant coefficients:

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad \text{for} \quad n \geq 1 \end{cases} \tag{2.14}$$

(OEIS: A000045) Fibonacci numbers:

$$\{0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, \ldots\}. \tag{2.15}$$

The generating function of the Fibonacci numbers is:

$$G_{\{F_n, n \geq 0\}}(x) = \frac{x}{1 - x - x^2} = \sum_{n=0}^{\infty} F_n x^n. \tag{2.16}$$

Rewriting the generating function as (which shows the form of the recurrence in the denominator):

$$G_{\{F_n, n \geq 0\}}(x) = \frac{\left(x^{-1}\right)^1}{\left(x^{-1}\right)^2 - \left(x^{-1}\right)^1 - \left(x^{-1}\right)^0}, \tag{2.17}$$

and setting x^{-1} to 10^k, we get the form:

$$\frac{\left(10^k\right)^1}{\left(10^k\right)^2 - \left(10^k\right)^1 - \left(10^k\right)^0} = \frac{10^k}{10^{2k} - 10^k - 1} = \sum_{n=0}^{\infty} \frac{F_n}{\left(10^k\right)^n}, \quad k \geq 1. \tag{2.18}$$

For example, for the first few values of k, we have (note that overlapping occurs when Fibonacci numbers have more than k digits):

$$\begin{aligned}
&k = 1 : 10/89 = 0.11235955056179775280898876404\ldots \\
&k = 2 : 100/9899 = 0.010102030508132134559046368320\ldots \\
&k = 3 : 1000/998999 = 0.001001002003005008013021034055 0\ldots \\
&k = 4 : 10000/99989999 = 0.000100010002000300050008001300 21\ldots
\end{aligned} \tag{2.19}$$

A variant of the above is:

$$\frac{1}{\left(10^k\right)^2 - \left(10^k\right)^1 - \left(10^k\right)^0} = \frac{1}{10^{2k} - 10^k - 1} = \sum_{n=0}^{\infty} \frac{F_n}{\left(10^k\right)^{n+1}}, \quad k \geq 1. \tag{2.20}$$

For example, for the first few values of k, we have (note that overlapping occurs when Fibonacci numbers have more than k digits):

$$\begin{aligned}
&k = 1 : 1/89 = 0.011235955056179775280898876404\ldots (\text{A021093})\\
&k = 2 : 1/9899 = 0.0001010203050813213455904636832 0\ldots\\
&k = 3 : 1/998999 = 0.000001001002003005008013021034055 0\ldots\\
&k = 4 : 1/99989999 = 0.000000010001000200030005000800130021\ldots
\end{aligned} \tag{2.21}$$

Definition 2.3.5. *Non-homogeneous linear recurrence relations with constant coefficients.*
An order k nonhomogeneous linear recurrence relation with constant coefficients is an equation of the form

$$\left(\sum_{i=0}^{k} c_i a(n-i)\right) + f(n) = (c_0 a(n) + c_1 a(n-1) + c_2 a(n-2) + \cdots + c_k a(n-k)) + f(n) = 0 \tag{2.22}$$

where $f(n) \neq 0$ and the k coefficients $c_i(\forall i)$ are constants.

Example. Non-homogeneous linear recurrences (of order 1) with constant coefficients:

$$\begin{cases} a_0 = 1 \\ a_n = 2a_{n-1} + 1, \quad n \geq 1 \end{cases} \tag{2.23}$$

$$\begin{cases} a_0 = 1 \\ a_n = 2a_{n-1} + n, \quad n \geq 1 \end{cases} \tag{2.24}$$

$$\begin{cases} a_0 = 1 \\ a_n = 2a_{n-1} + 2^n, \quad n \geq 1 \end{cases} \tag{2.25}$$

Example. Non-homogeneous linear recurrences (of order 2) with constant coefficients:

$$\begin{cases} a_0 = 1 \\ a_n = 3a_{n-1} + 2a_{n-2} + n, \quad n \geq 1 \end{cases} \tag{2.26}$$

Recurrence equation

When formulated as an equation to be solved, recurrence relations are known as recurrence equations or sometimes difference equations (i.e. a recurrence equation is the discrete analogue of a differential equation).

Definition 2.3.6. *A recurrence equation involves an integer function $f(n)$ in a form like:*

$$f(n) - f(n-1) = g(n) \tag{2.27}$$

where g is some integer function. The above equation is the discrete analog of the first-order ordinary differential equation:

$$f'(x) = g(x). \tag{2.28}$$

For the foregoing, a *linear recurrence equation* is a recurrence equation in a sequence of numbers $\{x_n\}$ expressing x_n as a first degree polynomial in x_k with $k < n$. For example:

$$x_n = Ax_{n-1} + Bx_{n-2} + Cx_{n-3} + \dots . \tag{2.29}$$

A quotient-difference table eventually yields a line of $0s$ iff the starting sequence is defined by a linear recurrence equation. The solutions to a linear recurrence equation can be computed straightforwardly, but quadratic recurrence equations are not so well understood.
The sequence generated by a recurrence relation is called a *recurrence sequence.* Let:

$$s(X) = \prod_{i=1}^{m} (1 - \alpha_i X)^{n_i} = 1 - s_1 X - \dots - s_n X^n \tag{2.30}$$

where the generalized power sum $a(h)$ for $h = 0, 1, \dots$ is given by:

$$a(h) = \sum_{i=1}^{m} A_i(h) \alpha_i^h \tag{2.31}$$

with distinct nonzero roots α_i, coefficients $A_i(h)$ which are polynomials of degree $n_i - 1$ for positive integers n_i, and $i \in [1, m]$. Then the sequence $\{a_h\}$ with $a_h = a(h)$ satisfies the recurrence relation:

$$xa_{h+n} = s_1 a_{h+n-1} + \dots + s_n a_h . \tag{2.32}$$

The terms in a general recurrence sequence belong to a finitely generated ring over the integers; therefore, it is impossible for every rational number to occur in any finitely generated recurrence sequence. If a recurrence sequence vanishes infinitely often, then it vanishes on an arithmetic progression with a common difference 1 that depends only on the roots. The number of values that a recurrence sequence can take on infinitely often is bounded by some integer l that depends only on the roots. There is no recurrence sequence in which each integer occurs infinitely often or in which every Gaussian integer occurs.

Generating function

When studying sequences and recurrences, it is often convenient to represent the sequence by a generating function. In mathematics, a generating function is a way of encoding an infinite sequence of numbers a_n by treating them as the coefficients of a formal power series. This series is called the generating function of the sequence. Unlike an ordinary series, the formal power series is not required to converge: in fact, the generating function is not actually regarded as a function, and the "variable" remains indeterminate.

Definition 2.3.7. *The ordinary generating function (OGF) of a sequence a_n is:*

$$G(a_n; x) = \sum_{n=0}^{\infty} a_n x^n . \tag{2.33}$$

When the term generating function is used without qualification, it is usually taken to mean an ordinary generating function.

Definition 2.3.8. *The exponential generating function (EGF) of a sequence a_n is:*

$$\mathrm{EG}\left(a_n; x\right) = \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}. \tag{2.34}$$

Exponential generating functions are generally more convenient than ordinary generating functions for combinatorial enumeration problems that involve labelled objects.
Example. The sequence 1, 3, 6, 10, 15, ... (OEIS: A000217) of triangular numbers are given by the following explicit formulas:

$$T_n = \sum_{k=1}^{n} k = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} = \frac{n^2+n}{2}, \binom{n+1}{2} \tag{2.35}$$

where $\binom{n+1}{2}$, does not mean division, but is the notation for a binomial coefficient. It represents the number of distinct pairs that can be selected from objects $n+1$, and is read aloud as "n plus one chooses two".

The generating functions of the sequence are:

$$\begin{aligned} G(x) &= \frac{1}{(1-x)^3}, \\ E(x) &= \left(1 + 2x + \frac{x^2}{2}\right) e^x. \end{aligned} \tag{2.36}$$

Generating functions provide a very efficient way to represent sequences.

Autocorrelation

Autocorrelation is a measure of the similarity between a sequence and a time-shifted replica of the sequence. Ideally, the autocorrelation function (ACF) should be impulsive, i.e. peak value at zero time shift and zero values at all other time shifts.

Crosscorrelation

Cross orrelation is the measure of similarity between two different sequences. The cross correlation between two sequences is the complex inner product of the first sequence with a shifted version of the second sequence, which indicates whether the two sequences are distinct. Ideally, it is desirable to have sequences with zero cross-correlation value at all time shifts.

Analysis

Various methodologies can be used to analyse integer sequences [Slo73] [SP95]. These include using a data compression algorithm, computing the discrete Fourier transform, or seeking a linear recurrence equation that links the terms or a generating function that produces them. Moreover, there exist a substantial number of transformations that establish a connection between different integer sequences. Such transformations include the Euler transform, the exponential transform, the Möbius transform, and others.

Chapter 3

Integer Sequences in cryptography

To conduct our research, we performed a thorough search-string inspection in multiple academic databases and search engines, with a primary focus on all available journals and conference proceedings. Our search strategy used a two-stage approach, with the first stage involving manual inspection using several search strings and their combinations using booleans. The databases and digital libraries we inspected within included ScienceDirect, Scopus, ACM Digital Library, Springer, IEEE Xplore, and Web of Science. In particular, all digital resources for publications from the International Association for Cryptologic Research (IACR) [iac] and the Journal of Integer Sequences [jis] were carefully searched. Moreover, we also used several web search engines, such as Google Scholar and Microsoft Academics, to access related literature across all publishing formats and disciplines.

During the first stage of our inspection, we conducted combinatorial searches using the search strings made up of various strings and terms within the combination of the above-mentioned booleans. We also tested the search strings on an iterative basis to fine-tune our search results, which enabled us to tackle the challenge of aligning our searches with completeness and consistency. The search strings were distinctively implemented in the title, abstract and corresponding keywords. In addition to this, we manually obtained several reputed journals and veteran conferences related to the domain of our research. These selected journals and conferences include previously held research in the field of cryptography. We also performed a second stage procedure to obtain a more significant sample of our research at the previous stage. During this stage, we scanned all reference lists and examined all of the main research to find additional articles. Our search strategy yielded a total of approximately 500 papers from all databases considered during the initial phase. After implementing our search strategy, we finally found a total of around 100 articles related to our relevant survey study. This fine-tuning of our search results and the use of multiple databases and search engines enabled us to conduct a comprehensive and rigorous analysis of the literature in our field of study.

3.1 Literature survey

3.1.1 Prime numbers

The history of prime numbers dates back to ancient times. Greek mathematicians, who were the first to study them extensively. The mathematicians of Pythagorean school (500 BC to 300 BC) were interested in numbers for their mystical and numerological properties, and they understood the idea of primality. The Rhind Mathematical Papyrus, from around 1550 BC, has Egyptian fraction expansions of different forms for prime and composite numbers. Euclid's Elements (c. 300 BC) proves the infinitude of primes and the fundamental theorem of arithmetic and shows how to construct a perfect number from a Mersenne prime. Eratosthenes, another Greek mathematician, created a screening method known as the Sieve of Eratosthenes, which allows all the prime numbers of a limited list to be identified by crossing multiples.

Interest in prime numbers was revived at the end of the Middle Ages. In the 17th century, the French monk Marin Mersenne defined the prime numbers that bear his name, obtained as $M_p = 2^p - 1$. Italian mathematician Pietro Cataldi had already shown in 1588 that $2^{19} - 1 = 524287$ is prime, setting a record for his time. In the 19th century, mathematicians such as Gauss and Riemann made further advances in the study of prime numbers, including the Prime Number Theorem. In the 20th century, computers gradually became important in calculating data for theorists to ponder. Since 1951, all the largest known primes have been found using tests on computers, such as the Lucas-Lehmer primality test. The search for ever-larger primes has generated interest and spurred the development of various branches of number theory, focussing on analytic or algebraic aspects of numbers.

Today, prime numbers are used in a variety of fields, including computer science, cryptography, and number theory.

Proprieties

Definition 3.1.1. *A number p is prime if (and only if) it is greater than 1 and has no positive divisors except for 1 and p.*

As is well known since primary school, by multiplying prime numbers, we can obtain in an essentially unique way each positive integer number. This is the gist of the Fundamental Theorem of Arithmetic, which Euclid already knew and expounded in Book VII of his Elements. Before discussing the Fundamental Theorem of Arithmetic, it is necessary to state the following simple but crucial characterisation of prime numbers.
OEIS. The first terms of the sequence are available in the OEIS database:

Proposition 3.1.1. *A positive integer number $p > 1$ is irreducible if and only if the following property holds: (P) whenever p divides a product ab, p divides a or b.*

Proof. We assume that p is irreducible and prove that (P) holds. So, assume that p divides ab and that p does not divide a. As p does not divide a, and as p does not have factors other than $\pm p$ and ± 1, we see that p and a have no non-trivial common factors, that is, $\gcd(a, p) = 1$. Therefore, there exist integers s and t such

A-number	A000040
Name	The prime numbers.
Data	$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, \cdots$
Offset	$1, 1$
Link	https://oeis.org/A000040

Table 3.1 Number primes in OEIS database.

that $1 = sa + tp$. By multiplying both sides by b, we obtain $b = sab + tpb$. As $p \mid ab$ and $p \mid p$, we may conclude that $p \mid b$. Vice versa, assume that (P) holds. If p were not prime, we would have $p = hk$ with h, k positive integers smaller than p. But we have $p \mid hk = p$, so $p \mid h$ or $p \mid k$, both of which are impossible because h and k are smaller than p. □

We are now in a position to prove the following.

Theorem 3.1.1. *(Fundamental Theorem of Arithmetic). Let n be an integer greater than 1. Then:*

$$n = p_1^{h_1} p_2^{h_2} p_3^{h_3} \cdots p_s^{h_s}, \tag{3.1}$$

where $p_1, p_2, \ldots, p_s$ are distinct prime numbers and the exponents h_j are positive, for all $j = 1, \ldots, s$. Furthermore, the representation (3.1) for n, called prime decomposition or factorisation of n, is unique up to the order of the factors.

Proof. We shall prove separately the existence and the uniqueness of the factorisation.

- **Existence of a factorisation**. The proof is by induction on the integer n to be factored. If $n = 2$, there is nothing to prove. Therefore, we may assume that the existence of a factorisation has been proved for each positive integer k with $2 \leq k < n$, and prove the same for n. If n is prime, there is again nothing to prove. Thus, let n be reducible, so it can be written as $n = ab$, with positive a and b, both greater than 1 and, therefore, smaller than n. Then, by the induction hypothesis, a and b are factorisable as products of primes:

$$a = p_1 p_2 \cdots p_r, \quad b = \bar{p}_1 \bar{p}_2 \cdots \bar{p}_s. \tag{3.2}$$

  Then,

$$n = p_1 p_2 \cdots p_r \bar{p}_1 \bar{p}_2 \cdots \bar{p}_s. \tag{3.3}$$

  Clearly, it suffices to group together, on the right-hand side, equal prime numbers in order to obtain the result in the form (3.1).

- **Uniqueness of the factorisation**. To prove the uniqueness of the factorisation for any integer n, we proceed by induction, this time on the number m of irreducible factors in any factorisation of n. Note that the number of factors appearing in the factorisation (3.1) is $m = h_1 + h_2 + \cdots + h_s$. If $m = 1$, then the number n having that factorisation is a prime number p. Assume that $n = p$ has another factorisation:

$$p = q_1^{k_1} q_2^{k_2} \cdots q_t^{k_t}. \tag{3.4}$$

As p is a prime that divides the right-hand side, it divides one of the factors of the right-hand side, for example $p \mid q_1$ (see Proposition 3.1.1). But q_1 is prime as well, so it has no non-trivial factors: hence $p = q_1$. By the cancelation property, which holds in $\mathbb{Z}$, we get:

$$1 = q_1^{k_1-1} q_2^{k_2} \cdots q_t^{k_t}. \tag{3.5}$$

This relation implies that all the exponents on the right-hand side are zero, otherwise we would have a product equal to 1 of integers greater than 1. Then the original right-hand side equals q_1, so $p = q_1$ is the only factorisation of n. So we have proved the basis of the induction. Assume now that the uniqueness of the factorisation has been proved for all integers that admit a factorisation into irreducible factors $m - 1$. Let n be an integer that admits a factorisation into m irreducible factors. Let us then:

$$n = p_1^{h_1} p_2^{h_2} \cdots p_s^{h_s} = q_1^{k_1} q_2^{k_2} \cdots q_t^{k_t} \tag{3.6}$$

be two factorisations of n into irreducible factors, the first consisting of m irreducible factors, that is, $h_1 + h_2 + \cdots + h_s = m$. Now, p_1 is a prime dividing the right-hand side, so it divides, for instance, q_1 (see again Proposition 3.1.1). As before, we have $p_1 = q_1$ and then, by the cancelation property, we get:

$$p_1^{h_1-1} p_2^{h_2} \cdots p_s^{h_s} = q_1^{k_1-1} q_2^{k_2} \cdots q_t^{k_t} \tag{3.7}$$

where the number of irreducible factors on the left-hand side is $m - 1$. By the induction hypothesis, in this case the uniqueness of the factorisation holds, so the primes q_j and the primes p_i are the same, up to the order. Then the factorisation of n is also unique.

□

The distribution of prime numbers. How many prime numbers are there? Euclid already knew the following theorem, which can be proved in several ways. We give here a completely elementary proof dating back to Euclid himself.

Corollary 3.1.1. *There are infinitely many prime numbers.*

Proof. Assume that the set of prime numbers is finite, consisting, for instance, of the numbers $p_1 < p_2 < \cdots < p_n$. Consider the number $N = p_1 \cdots p_n + 1$. This number is not prime, as it is greater than p_n, which, by hypothesis, is the greatest prime number. So N has a prime decomposition which can be written as:

$$N = p_1^{h_1} \cdots p_n^{h_n} \tag{3.8}$$

with at least one of the numbers $h_1, \ldots, h_n$ positive. Assume $h_i > 0$. Then $p_i \mid N$. Moreover, $p_i \mid (N - 1) = p_1 \cdots p_n$. Thus, $p_i \mid 1 = N - (N - 1)$, which is not possible, as $p_i > 1$. □

Recurrence relation. A possible formula using a recurrence relation is defined by:

$$a_n = a_{n-1} + \gcd(n, a_{n-1}), \quad a_1 = 7 \tag{3.9}$$

where $\gcd(x, y)$ denotes the greatest common divisor of x and y. The sequence of differences $a_{n+1}-a_n$ starts with $1, 1, 1, 5, 3, 1, 1, 1, 1, 11, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 23, 3, 1, \ldots$ (sequence OEIS: A132199). It is proven that this sequence contains only ones and prime numbers [Row08]. However, it does not contain all the prime numbers, since the terms $\gcd\,(n+1, a_n)$ are always odd and thus never equal to 2. 587 is the smallest prime (other than 2) not appearing in the first 10,000 results that are different from 1.
Nevertheless, it was conjectured that it contains all odd primes, even though it is rather inefficient. It is important to note that there is not a trivial programme that enumerates all and only prime numbers, as well as more efficient ones, so these recurrence relations are more curious than useful.
Functions. The distribution of primes among natural numbers has been intuited since antiquity, and it is well known that it is a random distribution. Consequently, functions that reliably generate prime numbers have often been acknowledged to have traction on the prime number set.
In order to obtain prime numbers, it is natural to ask for functions f defined for all natural numbers $n \geq 1$, which can be calculated in practice and produce some or all prime numbers. Several prime-generating functions can be classified into three classes for this purpose [Rib96]:

(a) $f(n) = p_n$; (the nth prime) for all $n \geq 1$.

(b) $f(n)$ is always a prime number, and if $n \neq m$, then $f(n) \neq f(m)$.

(c) The set of prime numbers is equal to the set of positive values assumed by the function.

In practice, these functions are generally impossible to compute. For example, both Gandhi's formula [pri71]:

$$p_n = \left\lfloor 1 - \log_2 \left(-\frac{1}{2} + \sum_{d \mid P_{n-1}} \frac{\mu(d)}{2^d - 1} \right) \right\rfloor \tag{3.10}$$

where $P_n = p_1 p_2 \cdots p_n$, and Willans' formula [Wil64]:

$$p_n = 1 + \sum_{i=1}^{2^n} \left\lfloor \left(\frac{n}{\sum_{j=1}^{i} \left\lfloor \left(\cos \frac{(j-1)!+1}{x} \pi \right)^2 \right\rfloor} \right)^{1/n} \right\rfloor \tag{3.11}$$

satisfy condition (a) but are essentially versions of the sieve of Eratosthenes [Gol74; GW67]. Gandhi's formula depends on properties of the Möbius function $\mu(d)$, while Willans' formula is based on Wilson's Theorem.

From a theoretical perspective, the functions satisfying (b) are interesting, even though all known members of this class are not practical prime generators. The first example proved the existence of a real number A such that $\lfloor A^{3^n} \rfloor$ is the prime for $n \geq 1$ (Mills' function). The only known way to find an approximation to a suitable A is by working backward from known large primes. Several relatives can be constructed similarly [Dud69].

The peculiar condition (c) is tailored to a class of multivariate polynomials constructed with this property [Mat71; JSWW76]. These results are implementations of primality tests in the language of polynomials, and thus they also cannot be used to generate primes in practise.

Applications to cryptography

Prime numbers play a crucial role in various cryptographic applications due to their unique mathematical properties. Here are some different uses of prime numbers in cryptography:

- **Cryptographic hash functions**
  - Prime numbers contribute to cryptographic hashing algorithms. They are used to create strong and unique hash values that are resistant to collision attacks. For example, SHA-2 is a family of cryptographic hash functions that produce a fixed length output from a variable length input [pri03]. In particular, SHA-2 uses prime numbers in several ways, to generate the initial values of the internal state, to generate the round constants, to generate the initial values of the truncated variants, to ensure the security of the message padding, which requires that the message length be a multiple of 512 bits.
- **Public-key cryptography**
  - RSA stands as a significant cryptographic algorithm in the realm of securing sensitive information across potentially insecure channels. Introduced by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977, RSA relies on the intricate properties of prime numbers to perform its encryption tasks [RSA78]. At its core, the fundamental premise of RSA is based on the complexity of factoring composite numbers into their prime components. This property sets the stage for its operation.
    RSA's security hinges on the formidable challenge of factoring the modulus, a semiprime resulting from the product of two prime numbers. This factorisation obstacle, especially with the inclusion of large prime numbers, renders the decryption process computationally formidable. In the contemporary landscape, RSA finds extensive use, acting as a linchpin for secure digital communication in various domains, including online transactions, digital signatures, and more. However, as computational capabilities grow, the need for longer key lengths arises to maintain a robust level of security.
  - Elliptic Curve Cryptography (ECC) represents a cryptographic methodology that draws on the intrinsic characteristics of elliptic curves and prime numbers to establish a robust layer of security for digital communication [Kob87; Mil86]. Unlike conventional approaches such as RSA and Diffie-Hellman, ECC operates by manipulating these elliptic curves, which are essentially sets of points that satisfy particular mathematical equations. Within the ECC, prime numbers are pivotal in shaping its

operational framework. Firstly, ECC operates within a "prime field", a finite mathematical space delineated by a prime number p. This field serves as the context within which mathematical operations like addition and multiplication take place on the chosen elliptic curves. This finite field, derived from prime numbers, forms the bedrock of ECC's security. Secondly, ECC generates cryptographic key pairs through a process similar to other encryption techniques. However, ECC distinguishes itself by utilising elliptic curves instead of modular exponentiation on integers. The heart of ECC's key generation process involves establishing a base point or "generator" on the elliptic curve. The private key emerges from multiplying this base point by a private integer, while the public key is the result of this base point being multiplied by the same integer.
In essence, ECC capitalises on the intricate properties of elliptic curves and prime numbers to create a cryptographic framework of increased security. By incorporating prime numbers into the formation of finite fields and key generation, the ECC ensures a strong mathematical foundation for secure communications and the implementation of advanced cryptographic algorithms.

- **Cryptographic protocols**
  - Diffie-Hellman Key-Exchange is a cryptographic method that allows two parties to establish a shared secret key, which can be used to secure their communication over an insecure channel. This protocol is named after its creators, Whitfield Diffie and Martin Hellman, and was introduced in 1976. The core of the Diffie-Hellman protocol is based on the mathematical challenge of calculating discrete logarithms within a finite field. The following is how the Diffie-Hellman Key-Exchange process works, along with the role of prime numbers in this context. To start, the two involved parties, Alice and Bob, need to agree on some public parameters. These parameters include a large prime number p and a generator modulo p, often referred to as g. These public values are known to both parties and can be communicated without risk. Next, both Alice and Bob select a private key. This private key is a number chosen randomly within a specific range. Using their private key, each calculates the corresponding public key. This public key is calculated by raising the generator g to the power of the private key and then taking the remainder when divided by the prime number p. The public keys are then exchanged between Alice and Bob. Once both parties have received each other's public keys, they can calculate a shared secret key. This is done by increasing the received public key to the power of their own private key and taking the rest when divided by the prime number p. Surprisingly, both Alice and Bob arrive at the same value for the secret key, even though they started with different private and public keys.
    The prime numbers play a critical role in this process, as we shall see below.

- **Implementation**

  - Implementation of the previous cryptographic primitives plays crucial roles in ensuring the security and integrity of Internet communication, authentication, email encryption, secure shell connections, VPNs, blockchain transactions, and the establishment of trust through certificate authorities.

Further details

Remark on Cryptography protocols. One of the primary objectives of cryptography is to establish secure communication channels between different parties. Security in this context encompasses various aspects such as data confidentiality, message integrity, and participant authentication. An approach to constructing secure channels is for the parties to share suitable cryptographic keys, which they maintain in strict confidence and use as inputs to encryption and message authentication algorithms. Typically, keys deemed appropriate for such cryptographic applications are required to possess the properties of length and randomness. This essentially ensures that they cannot be anticipated or thoroughly searched within a reasonable timeframe. We are now faced with the challenge of establishing these keys over an insecure network. Key-Exchange protocols are designed to address this issue. More formally, a Key-Exchange protocol is a cryptographic procedure in which two or more entities exchange messages to jointly determine a strong cryptographic key that cannot be computed by external parties.
In-depth analysis of sequence application. As we mentioned in the previous paragraph, Diffie-Hellman Key-Exchange is a fundamental cryptographic protocol that allows two parties to securely share a secret key over an insecure communication channel. This protocol relies heavily on the mathematical properties of prime numbers and generators within a finite cyclic group, typically involving modular arithmetic. The prime number p and the generator g play a critical role in ensuring the security and functionality of this cryptographic method, as it makes it difficult for attackers to deduce the secret key even if they know the public keys.

- **Prime number** p. The prime number p is a cornerstone of the Diffie-Hellman Key-Exchange. Its primary role is to define the finite field Z_p, which is the set of integers modulo p. This field has several important properties that are leveraged in the protocol:
  - *Finite field definition.* The prime p ensures that the set Z_p forms a finite field, which is a set of numbers with well-defined addition, subtraction, multiplication, and division operations (excluding division by zero). The finiteness of the field is important because it limits the number of possible values, making exhaustive search attacks computationally infeasible.
  - *Modular arithmetic.* Operations in the Diffie-Hellman Key-Exchange are performed modulo p. This modular arithmetic ensures that the results of these operations remain within the set Z_p, preventing overflow and maintaining the integrity of the computations.
  - *Security foundation.* The security of the Diffie-Hellman protocol is based on the difficulty of solving the discrete logarithm problem within the finite field Z_p. Specifically, given $g^a \bmod p$ and $g^b \bmod p$, it is computationally

challenging to determine a or b without knowing the other value. The choice of a large prime p makes this problem even more difficult, improving the security of Key-Exchange.

  – *Public parameter*. The prime p is a public parameter that is shared between the communicating parties. Both parties agree on this value before initiating the Key-Exchange process. The public nature of p does not compromise security, as the difficulty of the discrete logarithm problem remains regardless of the knowledge of p.

- **Generator** g. The generator g is another critical component of the Diffie-Hellman Key-Exchange. It is an element of the finite field Z_p that has the specific properties necessary for the protocol:

  – *Primitive root*. Ideally, the generator g is chosen to be a primitive root modulo p. A primitive root is an element whose powers generate all the non-zero elements of the field Z_p. In other words, g is a generator of the multiplicative group of integers modulo p, meaning that the set $\{g^1, g^2, \dots, g^{p-1}\} \bmod p$ contains all the elements from 1 to $p-1$. This property ensures that the key space is maximized, making it more difficult for an attacker to guess the secret key.

  – *Public parameter*. Like the prime p, the generator g is also a public parameter. Both parties agree on g before the Key-Exchange begins. The public nature of g does not reduce the security of the protocol because the security relies on the difficulty of the discrete logarithm problem.

  – *Exponential operations*. During the Key-Exchange, each party selects a private key (a random integer) and computes an exponential value using g. For example, if Alice chooses a private key a and Bob chooses a private key b, they compute $g^a \bmod p$ and $g^b \bmod p$, respectively. These values are then exchanged over the insecure channel. The use of g in these exponential operations is important because it ensures that the resulting values are uniformly distributed over the field Z_p, making it difficult for an attacker to predict the private keys.

  – *Shared secret computation*. After exchanging the exponential values, each party uses their private key to compute the shared secret. Alice computes $\left(g^b \bmod p\right)^a \bmod p$, and Bob computes $(g^a \bmod p)^b \bmod p$. Due to the properties of modular arithmetic, both computations yield the same result: $g^{ab} \bmod p$. This shared secret can then be used as a key for symmetric encryption algorithms to secure further communications.

Example of Diffie-Hellman Key-Exchange. To illustrate the roles of p and g in the Diffie-Hellman Key-Exchange, consider the following example:

– *Public parameters*. Alice and Bob agree on a large prime $p = 23$ and a generator $g = 5$.

– *Private keys*. Alice selects a private key $a = 6$, and Bob selects a private key $b = 15$.

– *Compute public values*:

  . Alice computes $A = g^a \bmod p = 5^6 \bmod 23 = 8$.

  . Bob computes $B = g^b \bmod p = 5^{15} \bmod 23 = 19$.

– *Exchange public values.* Alice and Bob exchange their computed public values A and B.

– *Compute shared secret*:

  . Alice computes the shared secret as $S = B^a \bmod p = 19^6 \bmod 23 = 2$.

  . Bob computes the shared secret as $S = A^b \bmod p = 8^{15} \bmod 23 = 2$.

Both Alice and Bob now share the secret value $S = 2$, which can be used for further secure communication. The prime number p and the generator g are fundamental to the security and functionality of the Diffie-Hellman Key-Exchange protocol. The prime p defines the finite field Z_p and ensures the security of the protocol through the difficulty of the discrete logarithm problem. The generator g acts as a primitive root and facilitates exponential operations that are essential to compute the shared secret. Together, these components enable two parties to securely exchange a secret key over an insecure channel, forming the basis for secure communication in many cryptographic systems.

3.1.2 Mersenne prime

The storey of Mersenne numbers begins in the 17th century with the French mathematician Marin Mersenne. Mersenne was a polymath who corresponded with many of the leading scientists and thinkers of his time, including René Descartes and Galileo Galilei.

Marin Mersenne introduced a special class of numbers that would later bear his name. These numbers took the form $2^n - 1$, where n is a positive integer. Mersenne's interest in these numbers was not merely theoretical; he believed that they had unique properties that made them suitable candidates for prime numbers.

Mersenne's contributions to the study of prime numbers extended beyond just naming these numbers. He developed what is now known as "Mersenne's test" to determine if a Mersenne number is prime. His test stated that if $2^n - 1$ is prime, then n must also be prime. This observation was a significant step forward in understanding prime numbers.

While Mersenne made strides in prime number theory, it was not until the 19th century that mathematicians like Édouard Lucas and Édouard Barbier began to explore the divisibility properties of Mersenne numbers in more detail. Their work laid the foundation for later research into these numbers.

The quest to find large prime numbers of the form $2^n - 1$ continued into the 20th century and beyond. With the advent of computers, it became possible to search and verify the primality of Mersenne numbers with exceptionally large values of n. The efforts of mathematicians and computer enthusiasts converged in projects like the Great Internet Mersenne Prime Search (GIMPS), founded by George Woltman in 1996.

GIMPS harnessed the power of distributed computing to systematically search for Mersenne primes. This collaborative effort led to the discovery of many record-breaking prime numbers, including some of the largest known primes.

In conclusion, the history of Mersenne numbers is a testament to the enduring fascination with prime numbers and the collaborative efforts of mathematicians and technology enthusiasts. These numbers, initially introduced by Marin Mersenne in the 17th century, continue to be a rich source of mathematical exploration and discovery in the modern era.

Proprieties

Definition 3.1.2. *A Mersenne prime is any prime number defined as $M_n = 2^n - 1$ where n is an integer.*

OEIS. The first terms of the sequence are available in the OEIS database:

A-number	A000668
Name	Mersenne primes (primes of the form $2^n - 1$).
Data	$3, 7, 31, 127, 8191, 131071, 524287, 2147483647, \cdots$
Offset	$1, 1$
Link	`https://oeis.org/A000668`

Table 3.2 Mersenne primes in OEIS database.

Numbers of the form $M_n = 2^n - 1$ without the primality requirement may be called simply Mersenne numbers (OEIS: A000225).

Connections to Perfect numbers. A positive integer n is called a perfect number if it is equal to the sum of all its positive divisors, excluding n itself. More than 2300 years ago, Euclid proved that if $2^k - 1$ is a prime number (it would be a Mersenne prime), then $2^{k-1}\left(2^k - 1\right)$ is a perfect number. A few hundred years ago Euler proved the converse (that every even perfect number has this form). It is still unknown whether there are any odd perfect numbers (but if there are, they are large and have many prime factors).

Theorem 3.1.2. *If $2^k - 1$ is a prime number, then $2^{k-1}\left(2^k - 1\right)$ is a perfect number and every even perfect number has this form.*

Proof. Suppose first that $p = 2^k - 1$ is a prime number and set $n = 2^{k-1}\left(2^k - 1\right)$. To show n is perfect, we need only show $\sigma(n) = 2n$. Since σ is multiplicative and $\sigma(p) = p + 1 = 2^k$, we know:

$$\sigma(n) = \sigma\left(2^{k-1}\right) \cdot \sigma(p) = \left(2^k - 1\right) 2^k = 2n. \tag{3.12}$$

This shows that n is a perfect number.
On the other hand, suppose n is any even perfect number and write n as $2^{k-1}m$ where m is an odd integer and $k \geq 2$. Again σ is multiplicative so:

$$\sigma\left(2^{k-1}m\right) = \sigma\left(2^{k-1}\right) \cdot \sigma(m) = \left(2^{k-1}\right) \cdot \sigma(m). \tag{3.13}$$

Since n is perfect we also know that:

$$\sigma(n) = 2n = 2^k m. \tag{3.14}$$

Together these two criteria give:

$$2^k m = \left(2^k - 1\right) \cdot \sigma(m) \tag{3.15}$$

so $2^k - 1$ divides $2^k m$ hence $2^k - 1$ divides m, say $m = \left(2^k - 1\right) M$. Now substitute this into the equation above and divide by $2^k - 1$ to get $2^k M = \sigma(m)$. Since m and M are both divisors of m we know that:

$$2^k M = \sigma(m) \geq m + M = 2^k M, \tag{3.16}$$

so $\sigma(m) = m + M$. This means that m is prime and its only two divisors are itself (m) and one (M). Thus, $m = 2^k - 1$ is a prime and we have proved that the number n has the prescribed form. □

Theorem 3.1.3. *If for some positive integer $n, 2^n - 1$ is prime, then so is n.*

Proof. Let r and s be positive integers, then the polynomial x^{rs-1} is x^{s-1} times $x^{s(r-1)} + x^{s(r-2)} + \ldots + x^s + 1$. So if n is composite (say rs with $1 < s < n$), then $2^n - 1$ is also composite (because it is divisible by 2^{s-1}). □

Notice that we can say more: suppose that $n > 1$. Since $x - 1$ divides $x^n - 1$, for the latter to be prime the former must be one. This gives the following.

Corollary 3.1.2. *Let a and n be integers greater than one. If $a^n - 1$ is prime, then a is 2 and n is prime*

Usually the first step in factoring numbers of the forms $a^n - 1$ (where a and n are positive integers) is to factor the polynomial $x^n - 1$. In this proof, we just used the most basic of such factorisation rules.
Lucas-Lehmer test. Let $M_p = 2^p - 1$ be the Mersenne number to test with p an odd prime. The primality of p can be efficiently checked with a simple algorithm such as trial division, since p is exponentially smaller than M_p. Define a sequence $\{s_i\}$ for all $i \geq 0$ by:

$$s_i = \begin{cases} 4 & \text{if } i = 0 \\ s_{i-1}^2 - 2 & \text{otherwise.} \end{cases} \tag{3.17}$$

The first few terms of this sequence are $4, 14, 194, 37634, \ldots$ (OEIS: A003010). Then M_p is prime if and only if:

$$s_{p-2} \equiv 0 \pmod{M_p}. \tag{3.18}$$

The number s_{p-2} mod M_p is called the Lucas-Lehmer residue of p. (Some authors equivalently set $s_1 = 4$ and test s_{p-1} mod M_p).
Example. The Mersenne number $M_3 = 2^3 - 1 = 7$ is prime. The Lucas-Lehmer test verifies this as follows. Initially s is set to 4 and then updated $3 - 2 = 1$ time:

$$s \leftarrow ((4 \times 4) - 2) \bmod 7 = 0. \tag{3.19}$$

Since the final value of s is 0 , the conclusion is that M_3 is prime.
On October 21, 2024, GIMPS discovered the largest known prime number, $2^{136,279,841}-1$, having 41,024,320 decimal digits. The new prime number, also known as M136279841, has been calculated using the Lucas-Lehmer primality test.
Open problems. It is not known whether the set of Mersenne primes is finite or infinite.

Applications to cryptography

The principal applications of Mersenne Prime numbers in cryptography are as follows:

- **Foundations**
  - A general-purpose pseudorandom number generator (PRNG), called Mersenne Twister, provides a 623-dimensional equidistribution up to 32-bit accuracy [MN98]. In this algorithm, a Mersenne prime period is used, which is achieved by modifying the previously proposed generators. For a n-bit word length, the Mersenne Twister generates integers in the range $[0, 2^n - 1]$. The algorithm is based on a matrix linear recurrence over a finite binary field $\mathbb{F}_2$. It is a twisted generalised feedback shift register of rational normal form, with state *bit* reflection and tempering. The basic idea is to define a series x_i through a simple recurrence relation, and then output numbers of the form x_i^T, where T is an invertible $\mathbb{F}_2$-matrix called a tempering matrix. The most commonly used version of the Mersenne Twister algorithm is based on the Mersenne prime $2^{19937}-1$. The standard implementation of that, MT19937, uses a 32-bit word length. There is another implementation (with five variants) that uses a 64-bit word length, MT19937-64 and generates a different sequence. Furthermore, the generator has an algorithm that is provided to check its primitivity and the computational complexity of this primitivity test is $\mathcal{O}(p^2)$, where p is the degree of the polynomial. However, it is not cryptographically secure, i.e. Cryptographically Secure Pseudorandom Number Generator (CSPRNG), unless the CryptMT variant is used. The reason is that observing a sufficient number of iterations (624 in the case of MT19937, since this is the size of the state vector from which future iterations are produced) allows one to predict all future iterations.
    Specifically, CryptMT is a stream cipher which is a combination of Linear Feedback Shift Register (LFSR) like Mersenne Twister and non-linear filter based on multiplication [MNHS05]. The period and high dimension of equidistribution as a stream cipher are theoretically assured. Moreover, it uses a booter to generate shorter sequence efficiently.
- **Cryptographic hash functions**
  - An hash function scheme called Hash Mersenne Number Transform (HMNT) based on a New Mersenne Number Transform (NMNT) [MD20]. The HMNT is defined as the modulo of the Mersenne numbers, where arithmetic operations are simple equivalents to ones' complement. It takes

an arbitrary length as input and generates a hash value with variable lengths (128, 256, and 512-bits or longer).

- **Public-key cryptography**
  - A public-key cryptosystem whose security is based on arithmetic modulo of Mersenne numbers [AJPS18]. These numbers have an extremely useful property. For any number x modulo p, and $y = 2^z$, where z is a positive integer, $x \cdot y$ is a cyclic shift of x by z positions and thus the Hamming weight of x is unchanged under multiplication by powers of 2 . The encryption scheme is based on the simple observation that, given a uniformly random n-bit string R, when we consider $T = F \cdot R + G (\bmod p)$, where the binary representation of F and G modulo p has low Hamming weight, then T looks pseudorandom, i.e., it is hard to obtain any non-trivial information about F, G from R, T. The public-key is chosen to be the pair (R, T), and the secret key is the string F. The encryption scheme also requires an efficient error correcting code with.

Further details

Remark on cryptographic Hash functions. Hash functions are cryptographic algorithms that transform an input of arbitrary length into a fixed-length output, known as a "digest" or "hash value", unique representation of the original input. They must have several key properties to be secure and effective, as their function is fundamental in many security and cryptographic applications due to the hash functions' ability to ensure:

- *Data integrity*. Hash functions are used to ensure the integrity of the data. If we have the hash of a message or file, we can compare it with the hash of a received message or file to verify that it has not been altered. This is commonly used in checksums and digital signatures.
- *Digital signatures*. A hash function is a crucial component in digital signatures. The hash of a document is calculated and then encrypted with the sender's private key. The recipient can decrypt the hash using the sender's public key and compare it to the hash calculated from the received document to verify its authenticity and integrity.
- *Password hashing*. Passwords are usually stored as hashes rather than as plain text. When a user enters a password, its hash is calculated and compared with the stored hash. This ensures that even if a password database is compromised, attackers cannot easily retrieve the original passwords.
- *Key Derivation Functions (KDFs)*. Hash functions are used to derive cryptographic keys from passwords or other sensitive information. KDFs transform an input of arbitrary length into fixed-length keys that can be used in encryption algorithms.
- *Security protocols*. Hash functions are used in various security protocols such as TLS, SSL, IPSec, and in technologies like Bitcoin and other cryptocurrencies to ensure the integrity and security of transactions and communications.

In-depth analysis of sequence application. As mentioned previously, from the Mersenne numbers it has been possible to construct a hash function scheme (HMNT) derived from NMNT [BH95]. The NMNT has proved to be an important Number-Theoretic Transform (NTT), which has been firmly recognised within the field of signal processing. Interesting applications of NTTs are found in the areas of digital filtering, image processing, fast coding and decoding, multiplication of large integers and matrices, deconvolution, and cryptography [Shp12].

The most recognised NTTs are the Fermat (FNT) [AB74] and Mersenne (MNT) [Rad72] number transforms. However, for standard signal processing applications the main drawback of these transforms is the stringent relationship between word length (the number of bits in the modulus), obtainable transform length, and a limited choice of possible word lengths. In order to retain the advantages of NTTs, the NMNT was consequently introduced, which alleviates this relationship. NMNT is defined modulo the Mersenne numbers, where arithmetic operations are simple equivalent to the complement of 1 and has the cyclic convolution property; therefore, it can be used for fast calculation of error-free convolutions and correlations [HB14]. The NMNT is a particularly interesting NTT as it has a long powers of two lengths up to 2^p, making it amenable to fast algorithms. However, NMNT can be used in one or several dimensions. Moreover, NMNT has several inherent advantages, such as its sensitivity to slight input variation, the long transform length, and variable block size [AGB10].

These properties can be exploited to design a hash function that is more secure and efficient.

Definition 3.1.3. *(Transform definition). The NMNT of an integer sequence $x(k)$ of length L is given by:*

$$X(k) = \left\langle \sum_{l=0}^{L-1} x(l)\beta(lk) \right\rangle_{Mp}, \quad k = 0, 1, 2, \ldots, L-1 \tag{3.20}$$

and its inverse has exactly the same form:

$$x(n) = \left\langle L^{-1} \sum_{k=0}^{L-1} X(k)\beta(nk) \right\rangle_{Mp}, \quad n = 0, 1, 2\ldots\ldots, L-1 \tag{3.21}$$

where:

$$\begin{aligned} \beta(lk) &= \beta_1(lk) + \beta_2(lk) \\ \beta_1(lk) &= \left\langle \mathrm{Re}\,(\alpha_1 + \alpha_2)^{lk} \right\rangle_{Mp} \\ \beta_2(lk) &= \left\langle \mathrm{Im}\,(\alpha_1 + j\alpha_2)^{lk} \right\rangle_{Mp} \end{aligned} \tag{3.22}$$

also:

$$\begin{aligned} &\alpha_1 = \pm \langle 2^q \rangle_{Mp} \\ &\alpha_2 = \pm \langle -3^q \rangle_{Mp} \\ &q = 2^{p-2} \\ &\langle\rangle_{Mp} \text{ represents modulo } M_p \end{aligned} \tag{3.23}$$

The values of α_1 and α_2 are of order $L = 2^{p+1}$. For transform length L/d where d is an integer power of two, β_1 and β_2 are given by:

$$\begin{aligned} \beta_1(lk) &= \left\langle \operatorname{Re}\left(\left((\alpha_1 + j\alpha_2)^d\right)^{lk}\right)\right\rangle_{Mp} \\ \beta_2(lk) &= \left\langle \operatorname{Im}\left(\left((\alpha_1 + j\alpha_2)^d\right)^{lk}\right)\right\rangle_{Mp} \end{aligned} \tag{3.24}$$

Re() and Im() denote real and imaginary parts of the enclosed term respectively, (L^{-1}) exists and is given by (2^{p-d}), where $L = 2^d$ and d is an integer, $0 \leqslant d \leqslant p$.

Calculating the transform parameters starts with choosing a prime number (p). The value of the prime number depends on the desired transform length and dynamic range. For example, for simplicity, choose a prime number $p = 7$. The modulus for the chosen prime is $M_p = 2^7 - 1 = 127$ and the maximum transform length, $L_{\max} = 128$.

Definition 3.1.4. *(NMNT cyclic convolution property). The NMNT has the cyclic convolution property; if $x(n)$ and $h(n)$ are two sequences to be convolved and $[y(n) = x(n) \circledast h(n)]$ is: the convolution result, then:*

$$Y(k) = X(k)\Gamma H(k) = X(k) \bullet H_{ev}(k) + X(N-k) \bullet H_{od}(k) \tag{3.25}$$

where $\circledast$ is the cyclic convolution operator and $\bullet$ is point-by-point multiplication.

$X(k), H(k)$ and $Y(k)$ stand for the NMNT transforms of $x(n), h(n)$ and $y(n)$ respectively. $H_{ev}(k)$ and $H_{od}(k)$ stand for even and odd parts of $H(k)$ respectively, which are given by:

$$\begin{aligned} H_{ev}(k) &= \langle (H(k) + H(N-k)) \times 2^{p-1} \rangle_{Mp} \\ H_{od}(k) &= \langle (H(k) - H(N-k)) \times 2^{p-1} \rangle_{Mp}\,. \end{aligned} \tag{3.26}$$

If both $x(n)$ and $h(n)$ are properly padded with zeros, their circular convolution given in (3.25) will be equivalent to their linear convolution. To avoid overflow, the modulus Mp must be chosen so that $y(n)$ does not exceed Mp, an upper bound is given by [AB75; BH95]:

$$|y(n)| \leqslant |x(n)|_{\max} \sum_{n=0}^{N-1} |h(n)| \leqslant Mp/2 \tag{3.27}$$

(**HMNT scheme**). An input message M of arbitrary length is required to generate a variable hash value H. Usually, HMNT supports three lengths of hash values, i.e. $H = 128, 256$ and 512 *bits* or longer. The HMNT process consists of the following steps:

- *Step 1.* Convert the input message M into the corresponding ASCII code value.
- *Step 2.* The original message M is divided into a number of blocks (m): $M = \{M_0, M_1, \dots, M_{m-1}\}$. The length of each block is denoted as n, where n is the length of the hash value. The shortage in the last block is padded with the equivalent number of space characters in the ASCII code, which is 32.

– *Step 3.* The secret key K is a series of characters that modify the input message M. These characters also convert into the corresponding ASCII code values. If the character length is less than the length of the hash value (n), the block is padded with the equivalent number of space characters in the ASCII code. Then, elements are added one by one to each block of the input message M.

– *Step 4.* Upon modifying the input message using the secret key K, NMNT (a formula that performs mathematical operations to transfer each block of the message to the transform domain) is applied to each block in the input message.

– *Step 5.* The final hash value H of the message M is obtained by summation (element-by-element addition) of transform output NMNT to each block.

3.1.3 Sophie German prime

Sophie Germain was born in Paris, France, in 1776, during a time when women were largely excluded from formal education and the world of mathematics. Despite these obstacles, her passion for mathematics led her to teach herself from the books in her family library. She adopted the alias "Monsieur LeBlanc" to correspond with some of the most prominent mathematicians of her era, including Carl Friedrich Gauss and Adrien-Marie Legendre.

One of Germain's early interests was in the field of number theory, and she focused her efforts on Fermat's Last Theorem, a famous problem that had puzzled mathematicians for centuries. Although she did not succeed in proving the theorem, her work was groundbreaking. She introduced the concept of "Fermat's Last Theorem for $n = p$," where p is a prime number. Around 1825, Sophie Germain proved, in fact, that the first case of Fermat's last theorem is true for such primes, i.e., if p is a Sophie Germain prime, then there do not exist integers x, y, and z different from 0 and none a multiple of p such that $x^p + y^p = z^p$. Her insights into this special case of the theorem laid the groundwork for future mathematicians who would eventually prove it, most notably Andrew Wiles in 1994.

However, Sophie Germain's most enduring legacy lies in her contributions to prime number theory. She became fascinated with prime numbers and, in particular, the study of primes of the form $2p + 1$, where both p and $2p + 1$ are prime. These special prime numbers, now known as "Sophie Germain primes," played a crucial role in her investigations. She developed theorems and relationships involving these primes, and her work was foundational for the development of modern number theory.

Despite facing significant gender-based discrimination and barriers to her mathematical pursuits, Sophie Germain's dedication to mathematics and her groundbreaking contributions to number theory continue to inspire mathematicians and serve as a testament to the power of determination and passion in the face of adversity. Her work opened doors for future generations of female mathematicians and remains a source of inspiration in the field.

Proprieties

Definition 3.1.5. *A prime number p is a Sophie Germain prime if $2p+1$ is also a prime.*

OEIS. The first terms of the sequence are available in the OEIS database:

A-number	A005384
Name	Sophie Germain primes p: $2p+1$ is also prime.
Data	$2, 3, 5, 11, 23, 29, 41, 53, 83, 89, 113, 131, 173, 179, 191, 233, 239, \cdots$
Offset	$1, 1$
Link	https://oeis.org/A005384

Table 3.3 Sophie Germain primes in OEIS database.

Safe prime. The number $2p+1$ associated with a Sophie Germain prime is called a safe prime. For example, 23 is a Sophie Germain prime and $2 \cdot 23 + 1 = 47$ is its associated safe prime.
Strong prime. A prime number q is a strong prime if $q+1$ and $q-1$ have a large prime factor (approximately 500 digits). For a safe prime value $q = 2\ p+1$, the number $q-1$ naturally has a large prime value, that is, p, so a safe prime value q meets part of the criteria for a strong prime. The execution time of certain methods of factoring a number with q as the primary factor depends in part on the size of the primary factor $q-1$. This is true, for example, with Pollard's $p-1$ algorithm.
Conjecture. It is conjectured that there are infinitely many Sophie Germain primes, although this has never been proven, and that the number of Sophie Germain primes up to x is

$$SG(n) \sim C\frac{n}{(\log n)^2} \tag{3.28}$$

where C is Hardy–Littlewood's twin prime constant.

$$C := 2\prod_{p>2}\frac{p(p-2)}{(p-1)^2} \approx 1.32032, \tag{3.29}$$

and the product is over all primes $p > 2$ [Sho09].

Applications to cryptography

The principal applications of Sophie German prime numbers in cryptography are as follows:

- **Foundations**
  - Sophie-Germain prime moduli are used in parallel Linear Congruential Generators (LCGs) for Monte Carlo simulations, providing an alternative to the commonly used Mersenne primes [MC04]. These primes, of the form $2^q - k$, where k can be as large as $\sqrt{2^q}$, are used as moduli in LCGs, and modular multiplication in an LCG with a Sophie-Germain prime modulus can be written in a specific equation. The choice of Sophie-Germain primes reduces initialisation time and provides competitive generation

time when appropriately chosen. The resulting Sophie-Germain prime modulus LCGs have been tested and compared to Mersenne primes.

- **Secret-key cryptography**
  - Mode called Sophie Germain Counter Mode (SGCM) has been proposed as a variant of the Galois/Counter Mode of operation for block ciphers. Instead of the binary field $\mathrm{GF}\left(2^{128}\right)$, it uses modular arithmetic in $\mathrm{GF}(p)$ where p is a safe prime $2^{128} + 12451$ with the corresponding Sophie Germain prime $\frac{p-1}{2} = 2^{127} + 6225$ [Saa11]. Although SGCM prevents the specific "weak key" attack, there are other ways to modify the message that will achieve the same forgery probability against SGCM as is possible against GCM: By modifying a valid n-word message, you can create an SGCM forgery with probability circa $\frac{n}{2^{128}}$. That is, its authentication bounds are no better than those of Galois/Counter Mode.
- **Public-key cryptography**
  - Safe and strong primes were useful as the factors of secret keys in the RSA cryptosystem [RS01; VZGS13], because they prevent the system being broken by some factorization algorithms such as Pollard's $p-1$ algorithm. However, with current factorisation technology, the advantage of using safe and strong primes appears to be negligible today.
- **Cryptographic protocols**
  - The issues about safe e strong primes apply in other cryptosystems as well, including Diffie-Hellman Key-Exchange and similar systems that depend on the security of the discrete log problem rather than on integer factorization [Che06]. If $2p+1$ is a safe prime, the multiplicative group of modulo integers $2p+1$ has a subgroup of high prime order. This prime order subgroup is usually desirable and the reason for using safe primes is so that the modulus is as small as possible relative to p. For this reason, the key generation protocols for these methods often depend on efficient algorithms to generate strong primes, which in turn depend on assumptions that these primes have sufficient density [Gor85].

Further details

Remark on Secret-key cryptography. Secret-key cryptography is usually classified as block ciphers or stream ciphers. In a block cipher, the plaintext is divided into fixed-sized chunks called blocks. A block is specified to be a bitstring (i.e. a string of 0's and 1's) of some fixed length (e.g., 64 or 128 bits). A block cipher will encrypt (or decrypt) one block at a time. In contrast, a stream cipher first uses the key to construct a keystream, which is a bitstring that has exactly the same length as the plaintext (the plaintext is a bitstring of arbitrary length). The encryption operation constructs the ciphertext as the exclusive-or of the plaintext and the keystream. Decryption is performed by computing the exclusive-or of the ciphertext and the keystream [Sti05].

In this regard, a block cipher mode of operation is an algorithm that uses a block cipher to provide information security such as confidentiality or authenticity [oSC]. A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block. Most modes require a unique binary sequence, often called an Initialization Vector (IV), for each encryption operation. The IV must be non-repeating, and for some modes must also be random. The IV is used to ensure that distinct ciphertexts are produced even when the same plaintext is encrypted multiple times independently with the same key. Historically, encryption modes have been extensively studied in regard to their error propagation properties in various scenarios of data modification. Later development regarded integrity protection as an entirely separate cryptographic goal. Some modern modes of operation combine confidentiality and authenticity in an efficient way, and are known as authenticated encryption modes.
One such mode is Galois/Counter Mode (GCM) which is widely adopted for its performance. GCM throughput rates for high-speed state-of-the-art communication channels can be achieved with inexpensive hardware resources [oSC07].
In-depth analysis of sequence application. *(Description of GHASH).* Let X be a concatenation of authenticated unencrypted data, CTR-encrypted ciphertext, and padding. This data is split into m 128-bit blocks X_i :

$$X = X_1 \| X_2 \| \cdots \| X_m \tag{3.30}$$

AES is used to derive the root authentication key $H = E_K(0)$. The same AES key K is also used as the data encryption key. In this case, we assume that H is unknown to the attacker as the scheme would otherwise be trivially breakable.

GHASH is based on operations in the finite field $GF\left(2^{128}\right)$. Horner's rule is used in this field to evaluate the polynomial Y:

$$Y_m = \sum_{i=1}^{m} X_i \times H^{m-i+1} \tag{3.31}$$

The authentication tag is $T = Y_m + E_K\left(IV \| 0^{31} \| 1\right)$, assuming that a 96-bit IV is used. The IV value must never be repeated as that would lead to the "forbidden attack" discussed by Joux in [oSC].
(SGCM). Mathematically, SGCM differs from GCM inly in the underlying field where GHASH's arithmetic operations are performed. While GCM uses the binary field $GF\left(2^{128}\right)$, SGCM uses traditional modular arithmetic in $GF(p)$, where:

$$p = 2^{128} + 12451 = 340282366920938463463374607431768223907 \tag{3.32}$$

Here $\frac{p-1}{2}$ is also a prime, a Sophie Germain prime.
All other aspects of SGCM are equivalent to GCM, except those described in the 'Multiplication operation on blocks' and the 'GHASH function' of NIST Special Publication 800-38D [oSC07].

3.1.4 Fibonacci numbers

Leonardo of Pisa, who later became known as Fibonacci, was born in Pisa, Italy, around 1170. He travelled extensively with his merchant father, which exposed him to various mathematical ideas from different cultures. During his travels, Fibonacci encountered the Hindu-Arabic numeral system, which was much more efficient for arithmetic calculations than the Roman numerals commonly used in Europe at the time.

Fibonacci was greatly impressed by the Hindu-Arabic numerals and wanted to introduce them to Europe. He realised that to do so, he needed to write a comprehensive book that would demonstrate the superiority of these numerals. This endeavour led him to write "Liber Abaci" in 1202, where he not only introduced the Hindu-Arabic numeral system but also included various mathematical topics.

In this influential book, Fibonacci discussed a problem that would eventually lead to the discovery of the Fibonacci sequence. The problem was related to the growth of a hypothetical rabbit population. He described a scenario in which a pair of rabbits produces another pair in their first month of life and then each subsequent month, they produce another pair, assuming that they never die.

The sequence of rabbit pairs that resulted from this scenario turned out to be the Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, and so on. Each number in the sequence represents the number of pairs of rabbits at the end of each month.

Fibonacci initially used the sequence to solve this rabbit population problem, but he soon realised its broader mathematical significance. He used the sequence to illustrate various mathematical concepts, including algebraic and geometric progressions.

Although Fibonacci did not name the sequence after himself, it became widely known as the "Fibonacci sequence" in his honour. The sequence gained popularity and importance over the centuries, and mathematicians like Leonardo Euler and Édouard Lucas made significant contributions to its study.

Today, the Fibonacci sequence is not only a mathematical curiosity but also a fundamental concept in mathematics and various fields such as art, architecture, and nature, where its mathematical properties are celebrated and explored.

Proprieties

Definition 3.1.6. *Fibonacci numbers can be defined by the recurrence relation:*

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad for \quad n \geq 1 \end{cases} \tag{3.33}$$

In some older definitions, the value $F_0 = 0$ is omitted, so that the sequence starts with $F_1 = F_2 = 1$, and the recurrence $F_n = F_{n-1} + F_{n-2}$ is valid for $n > 2$.
OEIS. The first terms of the sequence are available in the OEIS database:
Recurrence relation. Another recurrence relation for the Fibonacci numbers is:

$$F_{n+1} = \left\lfloor \frac{F_n(1+\sqrt{5})+1}{2} \right\rfloor = \left\lfloor \phi F_n + \frac{1}{2} \right\rfloor, \tag{3.34}$$

where $\lfloor x \rfloor$ is the floor function and ϕ is the "golden ratio" ($\phi = \frac{1}{2}(1+\sqrt{5}) =$

A-number	A000045
Name	Fibonacci numbers: $F(n) = F(n-1) + F(n-2)$ with $F(0) = 0$ and $F(1) = 1$.
Data	$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, \cdots$
Offset	$0, 4$
Link	https://oeis.org/A000045

Table 3.4 Fibonacci numbers in OEIS database.

1.618033988...). This expression follows from the more general recurrence relation:

$$\begin{vmatrix} F_{n+1} & F_{n+2} & \cdots & F_{n+k} \\ F_{n+k+1} & F_{n+k+2} & \cdots & F_{n+2k} \\ \vdots & \vdots & \ddots & \vdots \\ F_{n+k(k-1)+1} & F_{n+k(k-1)+2} & \cdots & F_{n+k^2} \end{vmatrix} = 0 \tag{3.35}$$

for $k > 2$. (The $k = 1$ case is trivially F_{n+1}, while the $k = 2$ case is essentially Cassini's identity and therefore equal to $(-1)^n$.)
Generating function. The generating function of the Fibonacci sequence is the power series:

$$s(z) = \sum_{k=0}^{\infty} F_k z^k = \sum_{k=1}^{\infty} F_k z^k = 0 + z + z^2 + 2z^3 + 3z^4 + \ldots \tag{3.36}$$

This series is convergent for any complex number z satisfying $|z| < 1/\varphi$, and its sum has a simple closed form:

$$s(z) = \frac{z}{1 - z - z^2}. \tag{3.37}$$

Proof. This can be proved by multiplying by $(1 - z - z^2)$:

$$\begin{aligned} \left(1 - z - z^2\right) s(z) &= \sum_{k=0}^{\infty} F_k z^k - \sum_{k=0}^{\infty} F_k z^{k+1} - \sum_{k=0}^{\infty} F_k z^{k+2} \\ &= \sum_{k=0}^{\infty} F_k z^k - \sum_{k=1}^{\infty} F_{k-1} z^k - \sum_{k=2}^{\infty} F_{k-2} z^k \\ &= 0z^0 + 1z^1 - 0z^1 + \sum_{k=2}^{\infty} \left(F_k - F_{k-1} - F_{k-2}\right) z^k \\ &= z \end{aligned} \tag{3.38}$$

where all terms involving z^k for $k \geq 2$ cancel out because of the defining Fibonacci recurrence relation. □

Applications to cryptography

The main applications of Fibonacci numbers in cryptography are as follows:

- **Foundations**

– A Lagged Fibonacci Generator (LFG) is an example of a pseudo-random number generator based on Fibonacci sequence. This class of random number generators is aimed at being an improvement on the "standard" linear congruential generator. Generalising the sequence (3.33)

$$S_n \equiv S_{n-r} \otimes S_{n-s} \pmod{m}, \quad 0 < r < s \tag{3.39}$$

the new term is some combination of any two previous terms. m is usually a power of 2 $\left(m = 2^M\right)$, often 2^{32} or 2^{64}. The $\otimes$ operator denotes a general binary operation. This may be either addition, subtraction, multiplication, or the bitwise exclusive-or operator (XOR). The theory of this type of generator is rather complex, and it may not be sufficient simply to choose random values for j and k. These generators also tend to be very sensitive to initialisation. Generators of this type employ k words of state (they "remember"' the last k values). If the operation used is addition, then the generator is described as an Additive Lagged Fibonacci Generator (ALFG) [MCPR95], if multiplication is used, it is a Multiplicative Lagged Fibonacci Generator (MLFG) [MS04], and if the XOR operation is used, it is called a Two-tap generalised feedback shift register (GFSR). The GFSR is also related to the Linear-Feedback Shift Register (LFSR) [Gol82; GK02].

– A variety of keystream generators have been suggested that are based on Fibonacci sequences, and at least one has been implemented. These generators are appealing because they can take advantage of the security results from the theory of shift register-based keystream generators, while running much faster in software [And95].

– Although LFSRs are one of the most popular devices for generating pseudo-random sequences, since they are simple, fast, and easy to implement in software and hardware, the main disadvantage is that in an LFSR, the current state is a linear function of the previous state, thus cryptographically insecure. As an alternative, a Non-Linear-Feedback Shift Register (NLFSR), Fibonacci based also, whose current state is a nonlinear function of its previous state can be used. More specifically, for an n-bit shift register r its next state is defined as:

$$r_{i+1}\left(b_0, b_1, b_2, \ldots, b_{n-1}\right) = r_i\left(b_1, b_2, \ldots, f\left(b_0, b_1, b_2, \ldots, b_{n-1}\right)\right), \tag{3.40}$$

where f is the non-linear feedback function. At present, the main application area of NLFSRs is cryptography [Zhi13]. The output sequences of NLFSRs are normally very hard to break with existing cryptanalytic methods.

– p-Fibonacci error-correcting codes are a type of error-correcting codes that are based on the Fibonacci p-sequence. They are defined as the numerical sequence $a_{p,n}$ given by the recursive relation $a_{p,n} = a_{p,n-1} + a_{p,n-p-1}$, with initial values $a_{p,1} = \ldots = a_{p,p+1} = 1$. For a given integer $p \geq 1$, the

p-Fibonacci matrix Q_p is a $(p+1) \times (p+1)$ matrix of the following form:

$$Q_p = \begin{bmatrix} 1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix}. \tag{3.41}$$

Fibonacci Q_p matrices allow us to define a method to encode and decode a message M and also to detect and correct errors that might occur in transmission of the encoding of M. These codes are used in cryptography to design identification protocols and quantum-resistant signature schemes [BMM21].

- **Cryptographic hash functions**
  - An audio fingerprint is a compact representation of an original signal and is considered as a short summary of an audio object. Therefore, the fingerprint is considered identification in the sense that it almost uniquely represents the signal. Audio fingerprinting can achieve the monitoring of audio content without metadata, which helps to identify an unknown audio clip from a database via the Internet, PC, microphone, mobile phone, etc. Furthermore, people also adopt audio fingerprinting technologies to protect the copyright of music, prohibit copyright infringement of songs, etc. A fast retrieval algorithm based on Fibonacci Hashing for the extension of the Philips's method to save memory and improve query speed [CXM+13].
  - Protein family classification is crucial for applications such as smart drug therapies, understanding protein functions, and building phylogenetic trees. Although sequencing techniques can reveal biological similarities between protein families, they are time-consuming. To address this challenge, a computer and artificial intelligence-based classification system has been developed. This system converts protein sequences into numerical representations. A novel protein mapping method based on Fibonacci Hashing assigns each amino acid code to Fibonacci numbers based on integer representations. These coded amino acids are then inserted into a hashing table for classification using recurrent neural networks [AT21].
- **Steganography**
  - Steganographic solution in which Fibonacci numbers play a crucial role in improving the capacity and security of data embedding in images [ACS06]. Using image decomposition based on the Fibonacci sequence, the algorithm can create a larger number of *bit* levels, specifically 12 levels compared to traditional 8. This increase in levels allows for a more efficient distribution of secret data, significantly improving the overall capacity of information embedding. In addition, the combination of Fibonacci

numbers with T-order statistics allows the algorithm to embed secret data in less obvious regions of the image. This adaptive approach minimises the visual impact of changes made during the embedding process, thus improving the resistance to detection by various steganalysis tools.

– In another approach, the *pixel* location choice is obtained by resorting to the p-Fibonacci series $F_p(i)$, where $F_p(i) = F_p(i-1) + F_p(i-p-1)$ and p is a non-negative integer that decides the sequence of values given to a singular series [Gow17]. Pixel positions are taken whose values are multiples of the numbers of $F_p(i)$ and embed information in them.

Further details

Remark on Pseudo-Random Number Generators. Pseudo-Random Number Generators (PRNGs) play a crucial role in cryptography for several reasons:

- *Key Generation.* PRNGs are used to generate cryptographic keys, which need to be random and unpredictable to ensure the security of the cryptographic system. A weak PRNG can lead to key predictability, making the system vulnerable to attacks.
- *Initialization Vectors (IV).* In many cryptographic algorithms, Initialization Vectors (IVs) are used to ensure that identical plaintexts produce different ciphertexts on encryption. PRNGs are often used to generate these IVs.
- *Nonces.* PRNGs generate nonces (numbers used once) that are crucial for preventing replay attacks where an attacker tries to reuse a previously captured set of messages.
- *Salts.* In password hashing, salts are random data added to passwords before hashing to ensure that even identical passwords have different hash results. PRNGs are used to generate these salts.
- *Stream Ciphers.* In stream ciphers, PRNGs generate a pseudorandom keystream that is XORed with plaintext to produce ciphertext. The security of stream ciphers is highly dependent on the strength and unpredictability of the PRNG.

In summary, PRNGs are fundamental to ensuring the randomness and security required in various cryptographic processes, and a compromise in PRNG quality can lead to severe vulnerabilities in the entire cryptographic system.
In-depth analysis of sequence application. In this context, LFGs have occupied a special place among such generators and have gained popularity for their ease of implementation and relatively low cost and complexity. For this reason, it has been used successfully in many situations since 1958 and it was a real shock to discover in the 1990s that they actually fail an extremely simple, non-contrived test for randomness. As anticipated in the previous paragraph, these generators are defined by their recurrence relation:

$$x_n = x_{n-r} \otimes x_{n-s} \bmod m. \tag{3.42}$$

The symbol $\otimes$ represents an operation that could be any of the following: addition ($+$), subtraction ($-$), multiplication ($\times$), or exclusive OR ($\oplus$) [Knu69]. To generate R bits random number, $m = 2^R$; r and s are called the lags of the generator where $r > s > 0$ [Alu97]. The output sequence is represented by x_n, and the time is denoted by n. See [LEMV17] for a more detailed analysis of this generator.

The maximum period that can be achieved by LFG depends on the specific operation employed, as illustrated in Table 3.5.

Operations	Maximum Attainable Period
Addition, mod	$p = 2^{N-1}(2^r - 1)$
Subtraction, mod	$p = 2^{N-1}(2^r - 1)$
Multiplication mod	$p = 2^{N-3}(2^r - 1)$
Exclusive-or	$p = 2^r - 1$

Table 3.5 Maximum attainable periods p of LFG $x_n = x_{n-r} \otimes x_{n-s} \bmod 2^m$

The term AFG is used to group generators that use the + or - operator. As shown in the above table, + and operators obtain large periods; therefore, most stringent statistical tests show that AFG produce satisfactory outcomes even if the lags have small values. As such, the equation of an AFG can be expressed as:

$$x_n = x_{n-r} \pm x_{n-s} \bmod m. \tag{3.43}$$

Where m denotes the base, r and s represent the lags of the past samples, and $\{x_0, \ldots, x_{r-1}\}$, constitute the seeds values [Alu97].

When $m = 2^N, N$ being the length of the word, and the trinomial $x^r + x^s + 1$ is irreducible and primitive over $\mathbb{F}_2$, the maximum period p is reached (subject to the condition that at least one of the seed values must be odd) and its value is $p = 2^{N-1}(2^r - 1)$.

As already mentioned, despite their advantages, LFG is subject to limitations regarding randomness and security. Therefore, in order to have sufficiently large periods and exhibit ideal random behaviour, large lags are required. However, large lags mean large amounts of memory, since the state of an LFG is proportional to its lags [MHK+21]

Recently, various studies have been done to modify LFG, due to its easy implementation and simplicity [MSH+22]. To overcome the aforementioned issues, a more sophisticated architecture for generating keys has been introduced in this work.

3.1.5 Lucas sequences

The storey of the Lucas sequence begins with its namesake, Édouard Lucas, a French mathematician born in 1842. Lucas was a prolific mathematician and number theorist who made significant contributions to various areas of mathematics. His interest in number sequences led him to the discovery and study of what would later be known as the Lucas sequence.

In the mid-19th century, Lucas started investigating sequences that shared similarities with the more famous Fibonacci sequence, which had been introduced to the western world by Leonardo of Pisa (Fibonacci) in his book Liber Abaci in

1202. Like Fibonacci, Lucas noticed that certain sequences of numbers exhibited a pattern in which each number was obtained by adding the two preceding ones.

Lucas decided to explore these sequences further and, in 1877, published a paper titled "Théorie des Nombres" where he introduced what we now call the Lucas sequence. He defined it as a sequence that starts with the numbers 2 and 1, just like Fibonacci, and then continues with each subsequent number being the sum of the two preceding ones. His goal was to study the properties and relationships of this sequence.

The sequence quickly gained attention in the mathematical community, and it was found to have various intriguing properties. It appeared in the study of number theory, particularly in the analysis of prime numbers. Lucas himself made significant contributions to the study of primes and was a pioneer in what would later be known as "Lucas sequences."

Over time, mathematicians and researchers continued to explore the properties of the Lucas sequence, uncovering its connections to many areas of mathematics, including algebra, combinatorics, and geometry. It also found applications in computer science, where it was used in algorithms and cryptography.

The Lucas sequence has even found its way into the natural world and art, with its mathematical patterns resembling some aspects of plant growth and aesthetics. This mathematical curiosity continues to captivate mathematicians and enthusiasts, serving as a testament to the enduring fascination with the beauty and elegance of mathematical patterns in our universe.

Proprieties

Definition 3.1.7. *The Lucas sequences $U_n(P,Q)$ and $V_n(P,Q)$ are constant-recursive integer sequences that satisfy the recurrence relation:*

$$L_n = P \cdot L_{n-1} - Q \cdot L_{n-2} \tag{3.44}$$

where P and Q are fixed integers [BW23].

Any sequence satisfying this recurrence relation can be represented as a linear combination of the Lucas sequences $U_n(P,Q)$ and $V_n(P,Q)$.
Recurrence relations. Given the previous two integer parameters P and Q, the Lucas sequences of the first kind $U_n = U_n(P,Q)$ $(n \in \mathbb{N})$ and of the second kind $V_n = V_n(P,Q)(n \in \mathbb{N})$ are defined by the recurrence relations [HZ10]:

$$\begin{aligned} U_0 = 0, \quad U_1 = 1, \quad &\text{and} \quad U_n = P \cdot U_{n-1} - Q \cdot U_{n-2}(n \geq 2), \\ V_0 = 2, \quad V_1 = p, \quad &\text{and} \quad V_n = P \cdot V_{n-1} - Q \cdot V_{n-2}(n \geq 2). \end{aligned} \tag{3.45}$$

The characteristic equation $x^2 - Px + Q = 0$ of the sequences U_n and V_n has two roots $\alpha = (P + \sqrt{D})/2$ and $\beta = (P - \sqrt{D})/2$ with the discriminant $D = P^2 - 4Q$. Note that $D^{1/2} = \alpha - \beta$. Furthermore, $D = 0$ means $x^2 - Px + Q = 0$ has the repeated root $\alpha = \beta = P/2$. It is well known that for any $n \in \mathbb{N}$ ([Rib96]),

$$P \cdot U_n + V_n = 2 \cdot U_{n+1}, \quad U_n = \frac{\alpha^n - \beta^n}{\alpha - \beta}, \quad V_n = \alpha^n + \beta^n. \tag{3.46}$$

More generally, the Lucas sequences $U_n(P,Q)$ and $V_n(P,Q)$ represent sequences of polynomials in P and Q with integer coefficients. Famous examples U_n and V_n can be considered as the generalisation of many integer sequences such as Fibonacci numbers, Mersenne numbers, Pell numbers, Lucas numbers, Jacobsthal numbers, and a superset of Fermat numbers (Table 3.6).

P	Q	$U_n(P,Q)$	$V_n(P,Q)$
1	-1	Fibonacci numbers (OEIS: A000045)	Lucas numbers (OEIS: A000032)
1	-2	Jacobsthal numbers (OEIS: A001045)	Jacobsthal–Lucas numbers (OEIS: A014551)
2	-1	Pell numbers (OEIS: A000129)	Pell–Lucas numbers (OEIS: A002203)
3	2	Mersenne numbers $2^n - 1$ (OEIS: A000225)	Numbers of the form $2^n + 1$, which include the Fermat numbers (OEIS: A000051)

Table 3.6 Some Lucas sequences generated for different values of P and Q

.

Generating functions. Ordinary generating functions are:

$$\sum_{n\geq 0} U_n(P,Q)z^n = \frac{z}{1 - Pz + Qz^2} \tag{3.47}$$

$$\sum_{n\geq 0} V_n(P,Q)z^n = \frac{2 - Pz}{1 - Pz + Qz^2} \tag{3.48}$$

Applications to cryptography

The principal applications of Lucas sequences in cryptography are as follows:

- **Foundations**
  - Probabilistic Lucas pseudoprime tests, which are part of the commonly used Baillie-PSW primality test [BW80]. The test defines Lucas pseudoprimes as follows: given integers P and Q, where $P > 0$ and $D = P^2 - 4Q$, let $U_k(P,Q)$ and $V_k(P,Q)$ be the corresponding Lucas sequences. Let n be a positive integer and let $\left(\frac{D}{n}\right)$ be the Jacobi symbol. We define $\delta(n) = n - \left(\frac{D}{n}\right)$. If n is a prime that does not divide Q, then the following congruence condition holds: $U_{\delta(n)} \equiv 0 \pmod{n}$. If this congruence does not hold, then n is not prime. If n is composite, then this congruence usually does not hold. These are the key facts that make Lucas sequences useful in primality testing.
  - Lucas sequences are used in some primality proof methods, including the Lucas-Lehmer-Riesel test, and the N + 1 and hybrid N − 1/N + 1 methods such as those in Brillhart-Lehmer-Selfridge 1975 [BLS75]. The algorithm is very similar to the Lucas-Lehmer test, but with a variable starting point depending on the value of k. Defined a sequence u_i for all $i > 0$

by: $u_i = u_{i-1}^2 - 2$. Then $N = k \cdot 2^n - 1$, with $k < 2^n$ is prime if and only if it divides u_{n-2}. The starting value u_0 is determined using the Lucas sequence term $V_k(P, 1)$ taken mod N.

- **Public-key cryptography**
  - Lucas sequence $V_n(P, Q)$ has been proposed to be used for public key cryptosystem (LUC), in a manner similar to the famous RSA, but using Lucas sequences modulo a composite number instead of exponentiation [SL93]. It has stipulated to have the same security level as RSA for the same size key, but is about twice as slow. However, many of the supposed security advantages of LUC over cryptosystems based on modular exponentiation are either not present, or not as substantial as claimed [BBL95]. Furthermore, the security of Lucas functions is polynomial-time equivalent to the generalized discrete logarithm problems [LTT95].
  - Similarly, Lucas sequences have been used in several discrete logarithm-based encryption schemes that have been proposed over time. In particular, a variant of a probabilistic public-key encryption scheme based on LUC [JHW05], another scheme based on quadratic fields quotients [Cas07], a novel algorithm for the computation of Lucas sequences is proposed to improve the efficiency of cryptosystems based on LUC [LLX$^+$12], a cryptosystem based on second order linear sequences in which semantic security is ensured [EF12]. Moreover, as linear sequences are not multiplicative, the main advantage of Lucas cryptosystems is that they are not formulated in terms of exponentiation.
- **Implementation**
  - An approach for anonymous multi-receiver public key encryption based on Lucas sequences and the Chinese Remainder Theorem, which provides secure transmission of messages to authorized receivers via insecure channels. The scheme is shown to be better against renowned attacks and prevailing anonymous multi-receiver algorithms through computational analysis [CEF19].
  - A cryptography method based on relationships of hyperbolic balancing and Lucas-balancing functions, as well as through the use of direct and inverse matrices, as well as the balancing matrices. The applying Strassen's method to improve the time complexity of solving equations involving the balancing matrix. This demonstrates that the use of matrices to represent and protect initial messages in the cryptography method [Ray20].
- **Attacks and cryptanalysis**
  - Lucas sequences were used for factoring RSA modules through an S-index formation as a comparative tool in the factoring process. The S-index pattern is used to design an algorithm to factor RSA modules by determining the quadratic residual on ciphertexts. Non-positional nature of Residue Number Systems (RNS) is used and compared with

the Comparative S-Index, which is equivalent to magnitude comparison in RNS [AALA16].

Further details

Remark on Public-key cryptography. Since the inception of the Internet, it has become customary to make use of public-key (or two-key) cryptography to secure Internet commerce. In such a scheme, each member of a group of individuals wishing to exchange information will have both a private key and a public key unique to that person. If Alice and Bob are members of this group and Alice wishes to communicate with Bob, she looks up his encryption key in a public directory and encrypts her message M to him using this key. On receiving this ciphertext, Bob uses his decryption key to decipher it and produce M. As an example of such a cryptosystem, consider the RSA system seen in Section 3.1.1. Each member of the group, say Bob (or a trusted authority acting on Bob's behalf), selects two large primes p and q of k digits at random, keeps them secret and computes $N = pq$. He also selects at random an integer $e(< N)$ such that $\gcd(e, \varphi(N)) = 1 (\varphi(N) = (p-1)(q-1))$ and solves the linear congruence:

$$ed \equiv 1 \pmod{\varphi(N)} \tag{3.49}$$

by the extended Euclidean algorithm to find d with $0 < d < \varphi(N)$. Bob's public encryption key is the pair (e, N), and his private decryption key is d. If Alice wishes to send a secure numerically encoded message $M(0 < M < N, \gcd(M, N) = 1)$ to Bob, she calculates $C \equiv M^e (\text{mod} N)(0 < C < N)$ and sends C to Bob. Bob can recover M from C by calculating:

$$C^d \equiv M^{ed} \equiv M^{1+t\varphi(N)} \equiv M \pmod{N}, \tag{3.50}$$

by Euler's theorem. Since $M < N$, it is uniquely determined. This scheme has been the subject of many cryptographic attacks, but with some modifications it has endured them all and is still widely used today. See Boneh [Bon99]. Of course, if an adversary can factor in the RSA modulus N, then he can break the system. However, in spite of the many improvements to integer factoring algorithms since the announcement of the RSA system in 1977, factoring N when $k = 1024$, say, seems still to lie in the distant future. Of course, this statement could become invalid should some group of clever individuals develop a new and better factoring algorithm or produce a universal quantum computer with a sufficient number of qubits. At this point, the latter scenario seems more likely. For information on quantum computers and computing, see Kaye et al. [KLM06].

One of the problems associated with the RSA cryptosystem is the process of selecting p and q. We note that because of the existence of factoring techniques $p-1$ and $p+1$, it is essential for the security of the system that each of the four numbers $p \pm 1$ and $q \pm 1$ has at least one large prime factor. The problem of producing such p and q was first examined by Williams and Schmid [WS79], where the use of the primality tests of [BLS75] was advocated. Since then, there have been further developments by Shawe-Taylor [ST86] and Maurer [Mau95]. The latter paper is a particularly valuable contribution to this problem.

In-depth analysis of sequence application. In 1993, Smith [Smi93] produced a public-key cryptosystem, called LUC, which was based on Lucas functions, as previously discussed. "The basic idea behind LUC is that of providing an alternative to RSA by substituting the calculation of Lucas functions for that of exponentiation. Although Lucas functions are somewhat more complex mathematically than exponentiation, they produce superior ciphers". However, the system has some important weaknesses, as pointed out by Bleichenbacher et al. [BBL95]. Nonetheless, it has since been recommended as a possible authentication system and continues to be the subject of active research. (See, e.g., Ibrahimpašić [Ibr09].) If we suppose that N is an RSA modulus and $\gcd(QD, N) = 1$, the basic idea behind LUC is the simple result, easily proved from the law of appearance, that $U_{m\psi(N)}(P,Q) \equiv 0 \pmod{N}$ and $V_{m\psi(N)}(P,Q) \equiv 2 \pmod{N}$, where m is any positive integer and $\psi(N) = (p^2-1)(q^2-1)$. In this scheme, Bob computes $\psi(N)$ and finds some positive $e < N$ at random such that $\gcd(e, \psi(N)) = 1$. As in RSA, the pair (e, N) will constitute his public key. He next solves the linear congruence:

$$ed \equiv 1 \pmod{\psi(N)} \tag{3.51}$$

for his private key d. (This is not exactly what is recommended in [Smi93], but as mentioned in [BBL95], it avoids the problem of message dependence).
For Alice to send a message $M(< N)$ to Bob, she first places $P = M$ and $Q = 1$ and computes $C \equiv V_e(M,1) \pmod{N}$ and sends C to Bob. Bob can recover M by computing $V_d(C,1) \equiv M \pmod{N}$. To see why this works, we observe that:

$$V_d(C,1) \equiv V_d(V_e(M,1),1) \equiv V_{ed}(M,1) = V_{1+t\psi(N)}(M,1) \equiv M \pmod{N} \tag{3.52}$$

The latter congruence follows from:

$$\begin{aligned} 2V_{1+t\psi(N)}(M,1) &= V_1(M,1)V_{t\psi(N)}(M,1) + DU_1(M,1)U_{t\psi(N)}(M,1) \\ &\equiv 2M \pmod{N}. \end{aligned} \tag{3.53}$$

The values of C and $V_d(C,1)$ can be quickly computed with appropriate techniques. We have mentioned that if an adversary can factor N, then the RSA scheme (and LUC) can be broken. This leaves the question of whether breaking RSA is equivalent in difficulty to factoring N. Boneh and Venkatesan [BV98] have provided evidence that suggests this is not the case, and Boneh and Durfee [BD00] have shown that if $d < N^{0.292}$, then N can be effectively factored. However, Aggarwal and Maurer [Mau95] have shown that breaking RSA is equivalent in difficulty to factoring the modulus under a generic model of computation, but this is a very restrictive model, as it does not exploit the *bit* representation of elements except for testing equality. However, there is a scheme somewhat similar to LUC for which it can be proved that breaking it is equivalent in difficulty to factoring N. This system makes use of the solutions of a certain Pell equation [JW09], which we have seen are essentially given by the Lucas functions. More information on this system can be found in Müller [Mul06].

3.1.6 Catalan numbers

The history of Catalan numbers is a fascinating journey that spans over 200 years, from their first discovery in the 18th century to modern times. Catalan numbers have a rich history of multiple rediscoveries and have become a fundamental concept in mathematics [Sta15].

– **History**. The storey begins with Ming Antu, a Chinese scientist and mathematician who wrote a book in the 1730s called "Quick Methods for Accurate Values of Circle Segments." Although the integrality of Catalan numbers did not play a role in Ming Antu's work, his book included trigonometric identities and power series that involved Catalan numbers. However, it was not until 1839 that the connection between Ming Antu's work and Catalan numbers was observed by Luo Jianjin.

  In 1751, Leonhard Euler, one of the most influential mathematicians in history, introduced and found a closed formula for Catalan numbers. Euler defined Catalan numbers as the number of triangulations of an $(n + 2)$-gon. He provided the values of Catalan numbers for $n \leq 8$ and observed a pattern in successive ratios. Euler guessed a formula for Catalan numbers and derived the generating function for them. Christian Goldbach and Johann Segner assisted Euler in his proof of the formula, and by 1759, a complete proof was obtained. Johann Andreas von Segner, another correspondent of Euler, played a significant role in the history of Catalan numbers. In the late 1750s, Euler suggested to Segner the problem of counting the number of triangulations of an n-gon. Segner accepted the challenge and found a recurrence relation for Catalan numbers. However, he made an arithmetic mistake in computing some values of Catalan numbers. Euler corrected the mistake and published Segner's paper with his own summary. The combination of Euler's and Segner's results, along with Goldbach's observation, provided a complete proof of the product formula for Catalan numbers.

  In 1766, Semen Kirillovich Kotelnikow, a Russian mathematician, wrote a paper elaborating on Catalan numbers. Although he claimed to have another way to verify the product formula for Catalan numbers, his work mainly involved playing around with the formula. Nicolas Fuss, Euler's assistant, introduced the Fuss-Catalan numbers and a generalisation of Segner's recurrence relation in 1795.

  In the 19th century, the French school of mathematicians made significant contributions to the study of Catalan numbers. Joseph Liouville, a French mathematician, received a question from Olry Terquem about deriving Euler's formula for Catalan numbers from Segner's recurrence. Liouville communicated this problem to various geometers, which led to a series of papers on Catalan numbers. Gabriel Lamé provided an elegant double-counting argument to derive Euler's formula. Eugene Charles Catalan obtained standard formulas for Catalan numbers and studied the problem of counting bracket sequences. Olinde Rodrigues, another mathematician, also made contributions to the study of Catalan numbers during this period.

– **Modern interpretations and Applications**. As mathematical research progressed, Catalan numbers found applications in numerous areas, including algebraic topology, graph theory, computer science, and more. Researchers and mathematicians discovered new combinatorial interpretations [Bru12; Gri06; Gri12; Kah13] and deeper connections between numbers and various mathematical phenomena. Stanley [Sta99] gave a list of 66 different combinatorial descriptions of Catalan numbers, and added some more to the list [Sta]. Some of the specific instances are as follows:

– The number of movements in xy-plane from $(0,0)$ to (n,n) with two kinds of moves:

$$R : (x,y) \to (x+1,y), \quad U : (x,y) \to (x,y+1), \tag{3.54}$$

such that the path never rises above the line $y = x$.

– Triangulations of a convex $(n+2)$-gon into n triangles by $n-1$ diagonals that do not intersect in their interiors.

– Binary parentheses of a string of $n+1$ letters.

– Binary trees with n vertices.

The solution to these problems is the nth Catalan number.

Proprieties

Definition 3.1.8. *The nth Catalan number can be expressed directly in terms of the central binomial coefficients by:*

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!} = \prod_{k=2}^{n} \frac{n+k}{k} \quad \textit{for } n \geq 0 \tag{3.55}$$

OEIS. The first terms of the sequence are available in the OEIS database:

A-number	A000108
Name	Catalan numbers: $C_n = \frac{1}{n+1}\binom{2n}{n}$
Data	$1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, \cdots$
Offset	$0, 3$
Link	`https://oeis.org/A000108`

Table 3.7 Catalan numbers in OEIS database.

An alternative expression for C_n is:

$$C_n = \binom{2n}{n} - \binom{2n}{n+1} \quad \text{for } n \geq 0 \tag{3.56}$$

which is equivalent to the expression given above because $\binom{2n}{n+1} = \frac{n}{n+1}\binom{2n}{n}$. This formulation shows that C_n is an integer, which is not immediately obvious from the first formula given. The above expression forms the basis for a proof of the correctness of the formula.

An additional representation expression is:

$$C_n = \frac{1}{2n+1}\binom{2n+1}{n} \tag{3.57}$$

which can be directly interpreted in terms of the cycle lemma.
Asymptotically, the Catalan numbers increase as:

$$C_n \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}, \tag{3.58}$$

in the sense that the quotient of the nth Catalan number and the expression on the right tends towards 1 as n approaches infinity.
Recurrence relation. Catalan numbers satisfy the recurrence relation:

$$C_{n+1} = C_0C_n + C_1C_{n-1} + \cdots + C_nC_0 = \sum_{k=0}^{n} C_kC_{n-k} \tag{3.59}$$

and

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \frac{2(2n+1)}{n+2}C_n. \tag{3.60}$$

Proof. There are several ways to prove (3.59), but perhaps the most elegant is by appealing to Dyck paths of length $2(n+1)$, which we saw above that C_{n+1} counts. Given a Dyck path of length $2(n+1)$, let $2(k+1)$ be the first non-zero coordinate x where the path hits the x-axis, then $0 \leq k \leq n$. The path is broken up into two pieces, the part on the left of $2(k+1)$ and the part on the right. The part to the right is a Dyck path of length $2(n-k)$, so it is counted as C_{n-k}. The part to the left is a north-east step, then a Dyck path of length $2k$, and then a south-east step. (The middle path is a Dyck path "on stilts"; it never dips below its starting point because it cannot hit the x-axis earlier than $2(k+1)$.) There are C_k of these. Therefore, there are a total of C_kC_{n-k} paths that hit the x-axis first at $2(k+1)$, and combining these terms gives C_{n+1}, which is the recurrence relation. □

Example. If $n+1=3$, then C_{n+1} counts the five Dyck paths pictured above:

- Path 1 has $k=2$, counted in C_2C_0.
- Path 2 has $k=2$, counted in C_2C_0.
- Path 3 has $k=1$, counted in C_1C_1.
- Path 4 has $k=0$, counted in C_0C_2.
- Path 5 has $k=0$, counted in C_0C_2.

The middle path of length 4 on paths 1 and 2 , and the top half of the left peak of path 3, are the Dyck paths on stilts referred to in the proof above.
Remark. Dyck Path is a lattice path in the coordinate plane that starts at the origin (0,0) and consists of steps in the positive x and y directions (up and right) with the following conditions:

– The path never goes below the x-axis (no steps below $y = 0$).

– The path ends on the line $y = 0$.

Generating function. The generating function for Catalan numbers is:

$$C_n = \sum_{n=0}^{\infty} C_n x^n = \frac{1-\sqrt{1-4x}}{2x} = \frac{2}{1+\sqrt{1-4x}}. \tag{3.61}$$

Given the recurrence (3.59), let us now just mention one aspect of generating functions, namely the binomial theorem for arbitrary exponents. When a is any complex number, or even an indeterminate, and $k \in \mathbb{N}$, then we define the binomial coefficient:

$$\binom{a}{k} = \frac{a(a-1)\cdots(a-k+1)}{k!}. \tag{3.62}$$

The "generalized binomial theorem" due to Isaac Newton asserts that:

$$(1+x)^a = \sum_{n\geq 0} \binom{a}{n} x^n. \tag{3.63}$$

This formula is just the formula for the Taylor series of $(1+x)^a$ at $x = 0$. For our purposes, we consider generating function formulas such as equation (3.63) to be "formal" identities. Questions of convergence are ignored.

Proof. Multiply the recurrence (3.59) by x^n and sum on $n \geq 0$. On the left-hand side we get:

$$\sum_{n\geq 0} C_{n+1} x^n = \frac{C(x)-1}{x}. \tag{3.64}$$

Since the coefficient of x^n in $C(x)^2$ is $\sum_{k=0}^{n} C_k C_{n-k}$, on the right-hand side we get $C(x)^2$. Thus:

$$\frac{C(x)-1}{x} = C(x)^2, \tag{3.65}$$

or

$$xC(x)^2 - C(x) + 1 = 0. \tag{3.66}$$

Solving this quadratic equation for $C(x)$ gives:

$$C(x) = \frac{1 \pm \sqrt{1-4x}}{2x}. \tag{3.67}$$

We have to determine the correct sign. Now, by the binomial theorem for the exponent $1/2$ (or by other methods),

$$\sqrt{1-4x} = 1 - 2x + \cdots. \tag{3.68}$$

If we take the plus sign in (3.67) we get:

$$\frac{1+(1-2x+\cdots)}{2x} = \frac{1}{x} - 1 + \cdots, \tag{3.69}$$

which is not correct. Hence, we must take the minus sign. As a check,

$$\frac{1-(1-2x+\cdots)}{2x} = 1+\cdots, \tag{3.70}$$

as desired.

From the generating function, it is easy to obtain the formula of C_n (3.61). ☐

Corollary 3.1.3. *We can now prove C_n in the Definition 3.1.8:*

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{n!(n+1)!}. \tag{3.71}$$

Proof. Consider the following:

$$\sqrt{1-4x} = (1-4x)^{1/2} = \sum_{n\geq 0}\binom{1/2}{n}x^n. \tag{3.72}$$

hence by (3.67),

$$\begin{aligned} C(x) &= \frac{1}{2x}\left(1-\sum_{n\geq 0}\binom{1/2}{n}(-4x)^n\right) \\ &= -\frac{1}{2}\sum_{n\geq 0}\binom{1/2}{n+1}(-4)^{n+1}x^n. \end{aligned} \tag{3.73}$$

Equating coefficients of x^x on both sides gives:

$$C_n = -\frac{1}{2}\binom{1/2}{n+1}(-4)^{n+1}. \tag{3.74}$$

It is routine to expand the right-hand side of equation (3.74) and verify that it is equal to $\frac{1}{n+1}\binom{2n}{n}$.

The expression $\frac{1}{n+1}\binom{2n}{n}$ is the standard way to write C_n explicitly. There is an equivalent expression that is sometimes more convenient:

$$C_n = \frac{1}{2n+1}\binom{2n+1}{n}. \tag{3.75}$$

Note also that:

$$C_n = \frac{1}{n}\binom{2n}{n-1}. \tag{3.76}$$

☐

Applications to cryptography

The principal applications of Catalan numbers in cryptography are as follows:

- **Secret-key cryptography**

  - Encryption and decryption procedure that involves dividing the message into blocks of m characters each, coding each character to its equivalent $8-bit$ binary number using the ASCII code table, and XORing the resulting binary number with the 8421 code of a decimal digit converted to a $4-bit$ binary number [KSC12]. The encrypted binary number is then coded back to text characters using the ASCII code table. This procedure is applied to all data blocks. The decryption procedure is the reverse of the encryption procedure.
  - Encryption and decryption method which aims to establish a secure communication channel between two entities [SAM$^+$21a]. It involves generating Catalan values that satisfy the Catalan number property, defining the Lattice Path movement space, and defining the key equalisation rules. Both sides generate a random value of an arbitrary length and encrypt it with their chosen Catalan-key. The encrypted values are used as the initial values of the first phase of the Maurer's protocol. Values A and B (key material) are completely equalised after the second phase and are usable for generating a symmetric cryptographic key through the final phase (privacy amplification) in Maurer's protocol. This neutralises the mutual information between Eve and the other participants that existed at the beginning of the protocol. Overall, the proposed procedure aims to provide a high level of security for the Key-Exchange process by using the specific properties of Catalan numbers and the Lattice Path combinatorics.

- **Public-key cryptography**
  - An elliptic curve encryption algorithm is proposed based on integer sequences of Catalan numbers [AKH13]. The proposed approach uses the Catalan sequence to generate secure keys, which are then used to encrypt the data. The set of points follows a secure key based on circular rotation, and the corresponding point that falls below the Catalan number is taken as the secured key noted K_i. The sequence and K are used to generate secure keys K_i. The addition operation is performed between the selected point and K_i to obtain a point Q_i, which is then used to form the encrypted block. Therefore, Catalan numbers are used to generate secure keys in the proposed ECC-based encryption algorithm.

- **Cryptographic protocols**
  - Secret key sharing protocol for establishing secure communication between two entities in smart city applications [SAM$^+$20]. The protocol is based on the properties of Fuss-Catalan numbers and Lattice Path combinatorics. The proposed scenario consists of three phases: generating a Fuss-Catalan object based on the grid dimension, defining the movement in the Lattice Path Grid, and defining the key equalisation rules. The authors present the security analysis of the protocol and its test in the experimental part. They also examine the equivalence of the proposed scenario with Maurer's satellite scenario and suggest a new scenario that implements an

information-theoretical protocol for public-key distribution. The paper concludes with proposed research directions on key management in smart city applications.

- **Implementation**
  - Encryption method based on Catalan objects and combinatorial structures with noncrossing or nonnested matching [SAM+21b]. The results showed that it is much more difficult to recognize ciphertext generated with the Catalan method than one made with the Data Encryption Standard (DES) algorithm. The paper also evaluated the quality of the generated Catalan key using statistical testing proposed by the National Institute of Standards and Technology. The proposed method has potential applications in e-Health IoT and smart cities data storage and processing.
  - Encrypting and decrypting files and plaintext using the Lattice Path method for combinatorial encryption [SAB18]. The plaintext is first converted into binary form and then a binary key is generated using the Catalan number formula. The binary key is used to determine which characters from the plaintext are transferred to the ciphertext. The encryption process involves selecting an ordered pair of 1 and 0 for each character in the plaintext, and transferring the character to the ciphertext only when its corresponding *bit* 1 gets its pair of *bit* 0. The decryption process is performed in reverse order of reading the binary key record, starting from the last *bit* and ending at the first *bit* in the key. The occurrence of *bit* 0 indicates an open pair and 1 closed pair. The number of possible valid paths in the network is directly determined by the calculating formula for the C_n set of Catalan numbers.
  - Procedure for generating cryptographic keys from a segment of a 3D image using a computational geometry algorithm [SSS18]. The generated keys are then used to encrypt and decrypt text based on the balanced parentheses combinatorial problem. The encryption process involves permuting the bits of the message using the generated key, while the decryption process involves reversing the permutation using the same key. The triangulation of the separated polygon is converted into a record that represents the Catalan key, which is then used in the encryption and decryption of text based on the balanced parentheses combinatorial problem. The authors also discuss the properties of Catalan keys and the number of valid values that satisfy the condition of balance for a given basis.
  - Encryption method based on Catalan random walks offers new possibilities for multimedia data protection to ensure the rights of participants in the multimedia distribution chain [SSA21]. The proposed encryption and decryption procedure consists of five phases: conversion, division, selection, encryption, and generation. In the conversion phase, the data (text or image) is loaded and converted into binary form. In the division phase, the binary sequence is divided into binary blocks of a specific size,

and the basis n is loaded for generating the set C_n, representing the set of Catalan objects, i.e., the set of keys. In the selection phase, the valid random walks through the binary block are generated using the Catalan key. In the encryption phase, the binary block is encrypted using the generated random walk. In the generation phase, the encrypted binary blocks are combined to form the encrypted multimedia content. The decryption procedure is the reverse of the encryption procedure, where the encrypted multimedia content is divided into binary blocks, and the valid random walks are generated using the Catalan key. The binary blocks are then decrypted using the generated random walk, and the decrypted binary blocks are combined to form the original multimedia content.

  – Application of Ballot Problem, Stack Permutations, and Balanced Parentheses in encryption and decryption of files and plaintext [SKB18]. These combinatorial problems are used to generate a secret key based on Catalan numbers, which is then used for encryption and decryption. The Ballot Problem is used to determine the number of combinations to put the $2n$ votes in such a way that in each adding a new vote, the number of votes that has been won by candidate A is greater than or equal to the number of the votes that candidate B has received. Stack Permutations are used to determine the number of ways to stack n distinct objects in a pile. Balanced Parentheses are used to determine the number of ways to arrange n pairs of parentheses such that they are balanced.

- **Steganography**
  – Novel data hiding method using Catalan numbers and Dyck words [SAM+19]. The data carrier retains its original shape and a GenerateStegoKey class is responsible for generating a complex stego-key that later allows retrieving a hidden message from the data carrier. The proposed encryption and decryption procedure involves selecting *bits* in the data carrier based on the sets n_i that generate the value C_n, which serves to generate Dyck Words. The set of additional parameters S, E, R reveals the initial and end position of *bits* and an additional condition. The initial position and end of the bits give a schedule in which they are taken, and the value D determines which bits are taken in the unchanged form and which are complemented. In the last step, a stego-key consisting of an ordered triple $K = n_i, S, E, R, D_i$. The security of the system is proven through state-of-the-art machine learning analysis.

  – Encryption process based on encoding an image in a binary record, converting a secret message (hidden information) into a binary record, and creating a Delaunay triangulation of a binary record of an image whose vertices are carriers of the secret message *bit* [SSSK21]. After that, by applying the stack permutation method and Catalan objects to the coordinates (x, y) of the Delaunay vertex, a completely new encrypted triangulation is obtained whose vertex coordinates (x, y) are placed in a sequence. The original image, an encrypted string with vertex coordinates

(x, y) in the form of Base64 code and an encrypted Delaunay triangulation, are sent to the user via the medium (Internet). Finally, in the process of decryption, by reapplying the Catalan object and the stack permutation method, the original Delaunay triangulation of the binary record of the image is created and the original information is revealed.

– An integer sequence named Catalan Transform (CT) has been exploited in the image steganography domain. At the outset, the cover image is decomposed into 2×2 non-overlapping blocks in row major order [MHGS21]. Then, each such block, that is, $4 - pixel$ group, is converted into the transform domain using CT. Secret *bits* are embedded in the transformed components in varying proportions, which facilitates us to achieve a payload in the range of 1 to 4 *bpp* (*bits* per *pixel*). Inverse Catalan Transform (ICT) is applied over transformed cum embedded quadruples to generate the stego-pixels in spatial domain. Successive embedding operation over an entire image ensures the formation of stego-image.

Further details

Remark on Steganography. A fundamental characteristic of image steganography is the assurance of secure communication during the transmission of stego-images across various networks or communication channels. Various methodologies for image steganography have been proposed, depending on the specific application and stages involved within the embedding process. Consequently, these systems can be categorised according to several criteria, including the type of cover-image used (either 2D or 3D images), the intended application, the retrieval process (whether reversible or irreversible), the nature of the embedding process (whether in the spatial or transform domain) and adaptive steganography.

The process of embedding secret data within the cover-image constitutes the fundamental mechanism of steganography. Given the potential for embedding secret data within both the spatial and transform domains of the cover-image, an intricate classification system predicated on the nature of the cover domain is employed.

Steganography in the transform domain involves embedding within the frequency domain. Before embedding within the spatial domain, the content is converted into the frequency domain. Upon completion of the embedding process, an inverse transform is executed to generate a steganographic image. Similarly, the extraction process requires execution in the frequency domain. Embedding in the frequency domain typically enhances the robustness, imperceptibility, and security of the embedded message. In the incipient phase of the development of steganography, the transform domain was not widely adopted because of significant distortion effects, despite offering greater message robustness in lossy channels. This occurs because the embedded bits are diffused over a broader area of the image, making it more suitable for robust watermarking. However, contemporary steganographic techniques continue to advance within the transform domain due to their inherent complexity and the potential to produce secure and robust steganographic outcomes [KPVH19; RAS$^+$23].

In-depth analysis of sequence application. In the transform domain employed in steganography, we find the Catalan Transform (CT). CT is one of the invertible transformations in the sequence of integers associated with Catalan numbers [Bar05]. Catalan transform has been used in image steganography and was only implemented in 2021 in research [MHGS21].

Proposition 3.1.2. *(Riordan matrix). The general term $T(n,k)$ of the Riordan matrix $(1, C(x))$ is given by*

$$T(n,k) = \sum_{j=0}^{k} \binom{k}{j}\binom{j/2}{n}(-1)^{n+j}2^{2n-k} \tag{3.77}$$

where $C(x)$ is defined in (3.67).

Proof. We seek $[x^n]\,(xC(x))^k$. To this end, we develop the term $(xC(x))^k$ as follows:

$$\begin{aligned}
x^k c(x)^k &= x^k\left(\frac{1-\sqrt{1-4x}}{2x}\right)^k \\
&= \frac{1}{2^k}(1-\sqrt{1-4x})^k \\
&= \frac{1}{2^k}\sum_{j=0}^{k}\binom{k}{j}(-\sqrt{1-4x})^j \\
&= \frac{1}{2^k}\sum_{j}\binom{k}{j}(-1)^j(1-4x)^{j/2} \\
&= \frac{1}{2^k}\sum_{j}\binom{k}{j}(-1)^j\sum_{i}\binom{j/2}{i}(-4x)^i \\
&= \frac{1}{2^k}\sum_{j}\binom{k}{j}(-1)^j\sum_{i}\binom{j/2}{i}(-4)^i x^i \\
&= \sum_{j}\binom{k}{j}\sum_{i}\binom{j/2}{i}(-1)^{i+j}2^{2i-k}x^i.
\end{aligned} \tag{3.78}$$

Thus $[x^n]\,(xC(x))^k = \sum_{j=0}^{k}\binom{k}{j}\binom{j/2}{n}(-1)^{n+j}2^{2n-k}$. □

Definition 3.1.9. *(Catalan transform). Given a sequence a_n, its Catalan transform b_n is given by:*

$$\begin{aligned}
b_n &= \sum_{k=0}^{n}\frac{k}{2n-k}\binom{2n-k}{n-k}a_k \\
&= \sum_{k=0}^{n}\frac{k}{n}\binom{2n-k-1}{n-k}a_k
\end{aligned} \tag{3.79}$$

or

$$b_n = \sum_{j=0}^{n}\sum_{k=0}^{n}\frac{2k+1}{n+k+1}(-1)^{k-j}\binom{2n}{n-k}\binom{k}{j}a_j. \tag{3.80}$$

The inverse transformation is given by:

$$\begin{aligned} a_n &= \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-k}{k} (-1)^k b_{n-k} \\ &= \sum_{k=0}^{n} \binom{k}{n-k} (-1)^{n-k} b_k. \end{aligned} \tag{3.81}$$

Proof. Using [Spr04] we have:

$$\begin{aligned} T(n,k) &= 2^{2n-k}(-1)^n \sum_{j=0}^{k} \binom{k}{j} \binom{\frac{j}{2}}{n} (-1)^j \\ &= 2^{2n-k}(-1)^n \left\{ (-1)^n 2^{k-2n} \left(\binom{2n-k-1}{n-1} - \binom{2n-k-1}{n} \right) \right\} \\ &= \binom{2n-k-1}{n-1} - \binom{2n-k-1}{n} \\ &= \frac{k}{n} \binom{2n-k-1}{n-1} = \frac{k}{n} \binom{2n-k-1}{n-k}. \end{aligned} \tag{3.82}$$

But:

$$\begin{aligned} \frac{k}{n} \binom{2n-k-1}{n-k} &= \frac{k}{n} \frac{2n-k-(n-k)}{2n-k} \binom{2n-k}{n-k} \\ &= \frac{k}{n} \frac{2n-k-n+k}{2n-k} \binom{2n-k}{n-k} \\ &= \frac{k}{2n-k} \binom{2n-k}{n-k}. \end{aligned} \tag{3.83}$$

This proves the first two assertions of the proposition. The last assertion follows from the expression for the general term of the matrix $(1, x(1-x))$ obtained above. The equivalence of this and the accompanying expression is easily obtained. □

(Catalan transform-based secured image steganography). The application of CT in image steganography is innovative, with its most prominent characteristic being the significant variations in transformed coefficients caused by embedding. This is due to the unique computational process of the coefficients, making the scheme effectively impervious to adversarial attacks. Furthermore, the ingenuity of the approach is further highlighted by the precise manipulation of these transformed coefficients through an adaptive embedding rule, which guarantees a high payload capacity, improved perceptual quality and, critically, the ease of data extraction by the recipient without compromising the embedding capacity [MHGS21].

Let us consider $p_0, p_1, \ldots, p_n$ be the *pixel* values in a given *pixel* group P. Applying CT[10] one can compute the transformed components $t_0, t_1, \ldots, t_n$ as follows:

$$t_n = \sum_{k=0}^{n} \frac{k}{2n-k} \binom{2n-k}{n-k} p_k \tag{3.84}$$

where, for all $n, 0 \leq n \leq \text{size}(P)-1$. By using (3.84), CT is applied over $4-$*pixel* groups (i.e. size $(P) = 4$) to derive transformed quadruples as follows:

$$t_i = \begin{cases} p_0 & \text{if } i = 0 \\ p_1 & \text{if } i = 1 \\ p_1 + p_2 & \text{if } i = 2 \\ 2p_1 + 2p_2 + p_3 & \text{if } i = 3 \end{cases} \tag{3.85}$$

Again, by applying inverse CT (ICT), one can recalculate *pixel* values $p'_0, p'_1, \ldots, p'_n$ as:

$$p'_n = \sum_{k=0}^{n} (-1)^{(n-k)} \left(\frac{k}{n-k}\right) t_k \tag{3.86}$$

where, for all $n, 0 \leq n \leq \text{size}(P)-1$. Using (3.86), ICT is applied over transformed quadruples to recompute the $4-$*pixel* groups as follows:

$$p'_i = \begin{cases} t_0 & \text{if } i = 0 \\ t_1 & \text{if } i = 1 \\ -t_1 + t_2 & \text{if } i = 2 \\ -2t_2 + t_3 & \text{if } i = 3 \end{cases} \tag{3.87}$$

In case of no embedding, all recomputed *pixel* values are found to be exactly the same corresponding to the *pixel* values used prior to applying CT, i.e., $p'_i = p_i$.

The advantages of transform domain approaches and the application of CT have been previously mentioned. CT is utilised to convert each 2×2 non-overlapping block of the cover image from the spatial domain into the transformed domain. The core principle of CT involves the creation of an integer polynomial sequence in coefficient form through additions and multiplications based on the *pixel* values. Instead of embedding the secret bits directly into the *pixel* values, they are inserted into the transformed coefficients, thereby achieving increased robustness against typical signal processing attacks. Additionally, the extraction phase is employed to retrieve the secret image from the stego-image.
Specifically, the method is divided into two distinct phases that yield the following outcomes:

– The cover-image of dimension $m \times n$ is decomposed into 2×2 nonoverlapping blocks where the *pixel* values p_0, p_1, p_2 and p_3 are arranged as one dimensional sequence named as $4-$*pixel* group. CT is then applied to each *pixel* group to convert the same to the transform domain. Secret *bit*-stream s is obtained from the secret image. To achieve an average payload of n *bpp*, $4n$ numbers of *bits* from secret *bit* stream s (as obtained from secret image) are embedded in the transformed quadruples (t_0, t_1, t_2 and t_3). ICT is applied over the transformed cum embedded quadruples to get back Stego-pixels in the spatial domain. The above steps are repeated until and unless the secret bit-stream gets fully embedded and the stego-image is produced.

– The stego-image of dimension $m \times n$ is decomposed into 2×2 non-overlapping blocks where the stego-pixels p', p'_1, p'_2 and p'_3 are arranged as a one-dimensional sequence named group $4-$*pixel*. CT is then applied on each group of *pixels* to convert the same into the transform domain. For an average payload of n *bpp*, $4n$ numbers of secret bits are extracted from the transformed quadruples. ICT is applied to get the $4-pixel$ group back in the spatial domain. This process is repeated until and unless the entire secret *bit*-stream is extracted and the secret image is reproduced.

3.1.7 Narayana numbers

In 1356, the Indian mathematician Narayana Pandit wrote his famous book titled Ganita Kaumudi where he proposed the following problem of a herd of cows and calves: *A cow produces one calf every year. Beginning in its fourth year, each calf produces one calf at the beginning of each year. How many calves are there altogether after 20 years?* [AJ96] [Goy18]

We can translate this problem into our modern language of recurrence sequences. In this problem, we observe that the number of cows increased by one after one year, increased by one after two years, increased by one after three years, increased by two after four years, and so on. Hence we obtain the sequence $1, 1, 1, 2, \ldots$ in the n-th year [BDG20]. This problem appears to be similar to the Fibonacci rabbit problem discussed earlier.

The Narayana sequence has a close connection to some famous numbers or sequences and plays an important role in cryptography and combinatorics [RS15]. For example, it can be seen as the number of compositions of n in parts 1 and 3. For $n \geq 3$, the Narayana sequence can be expressed as the sums of rows of Pascal's triangle with triplicated diagonals, while the Fibonacci number F_n is the row sums of Pascal's triangle with slope diagonals of 45 degrees. Narayana's sequence has a beautiful distribution pattern, the ratio of consecutive terms whose consecutive terms approximate the super-golden ratio, which is closely related to the golden ratio [Siv20]. Moreover, the Narayana sequence satisfies good cross-correlation and autocorrelation properties, which provide wide applications in data coding, information theory, and cryptography [Lin21].

Proprieties

Definition 3.1.10. *Narayana's cow sequence can be defined by the recurrence relation:*

$$\begin{cases} N_0 = 0 \\ N_1 = N_2 = 1 \\ N_n = N_{n-1} + N_{n-3} \quad \textit{for} \quad n \geq 3 \end{cases} \tag{3.88}$$

a natural variation on the Fibonacci sequence.

OEIS. The first terms of the sequence are available in the OEIS database:
Recurrence relation. Narayana's cows sequence satisfies a third-order recurrence: relation

$$N_n = N_{n-1} + N_{n-3}, \text{ for } n \geq 3. \tag{3.89}$$

This has the initial values $N_0 = 0$ and $N_1 = N_2 = N_3 = 1$. Explicitly, the characteristic equation of N_n is:

$$x^3 - x^2 - 1 = 0 \tag{3.90}$$

A-number	A000930
Name	Narayana's cows sequence: $a(0) = a(1) = a(2) = 1$; thereafter $a(n) = a(n-1) + a(n-3)$.
Data	$1, 1, 1, 2, 3, 4, 6, 9, 13, 19, 28, 41, 60, 88, 129, 189, 277, 406, 595, 872, \cdots$
Offset	$0, 4$
Link	https://oeis.org/A000930

Table 3.8 Narayana sequence in OEIS database.

and the characteristic roots are:

$$\begin{aligned} \alpha &= \frac{1}{3}\left(\sqrt[3]{\frac{1}{2}(29-3\sqrt{93})} + \sqrt[3]{\frac{1}{2}(3\sqrt{93}+29)} + 1\right) \\ \beta &= \frac{1}{3} - \frac{1}{6}(1-i\sqrt{3})\sqrt[3]{\frac{1}{2}(29-3\sqrt{93})} - \frac{1}{6}(1+i\sqrt{3})\sqrt[3]{\frac{1}{2}(3\sqrt{93}+29)} \\ \gamma &= \frac{1}{3} - \frac{1}{6}(1+i\sqrt{3})\sqrt[3]{\frac{1}{2}(29-3\sqrt{93})} - \frac{1}{6}(1-i\sqrt{3})\sqrt[3]{\frac{1}{2}(3\sqrt{93}+29)} \end{aligned} \tag{3.91}$$

Then, the Narayana sequence can be obtained by Binet's formula:

$$\mathrm{N}_n = A\alpha^n + B\beta^n + C\gamma^n. \tag{3.92}$$

Generating function. For $n \in \mathbb{Z}_{\geq 0}$, the generating function of the Narayana sequence is:

$$g(x) = \frac{1}{1-x-x^3} = \sum_{n=0}^{\infty} N_{n+1}x^n. \tag{3.93}$$

With the Vieta theorem, we have:

$$\begin{cases} \alpha+\beta+\gamma = 1 \\ \alpha\beta+\beta\gamma+\alpha\gamma = 0 \\ \alpha\beta\gamma = 1 \end{cases} \tag{3.94}$$

From (3.92), we obtain:

$$\begin{aligned} N_0 &= A+B+C = 0 \\ N_1 &= A\alpha+B\beta+C\gamma = 1 \\ N_2 &= A\alpha^2+B\beta^2+C\gamma^2 = 1 \end{aligned} \tag{3.95}$$

which implies:

$$A = \frac{1-\beta-\gamma}{(\alpha-\beta)(\alpha-\gamma)}, \quad B = \frac{1-\alpha-\gamma}{(\beta-\alpha)(\beta-\gamma)}, \quad C = \frac{1-\alpha-\beta}{(\gamma-\beta)(\gamma-\alpha)}. \tag{3.96}$$

With formula (3.94), we can simplify A, B, and C and obtain:

$$A = \frac{\alpha}{\alpha^2-\alpha\beta-\alpha\gamma+\beta\gamma} = \frac{\alpha}{\alpha^2+2\beta\gamma} = \frac{\alpha^2}{\alpha^3+2} \tag{3.97}$$

and

$$B = \frac{\beta^2}{\beta^3+2}, \quad C = \frac{\gamma^2}{\gamma^3+2}. \tag{3.98}$$

The Narayana sequence was originally defined at positive indices. Actually, it can be extended to negative indices by defining:

$$N_{-n} = \frac{A}{\alpha^n} + \frac{B}{\beta^n} + \frac{C}{\gamma^n} \tag{3.99}$$

The following recurrence relation holds for all integral indices:

$$N_n = N_{n-1} + N_{n-3}, \quad n \in \mathbb{Z}. \tag{3.100}$$

Through a simple computation, the first few terms of N_n at negative indices can be obtained from formulas (3.98) and (3.99), so that $N_{-1} = 0, N_{-2} = 1, N_{-3} = 0, N_{-4} = -1$, which also satisfies relation (3.100).

Corollary 3.1.4. *(Limiting Ratio). Assuming that the ratio of the consecutive terms of the Narayana cow sequence described in (3.88) and through the recursive relation (3.89) is constant, we shall try to determine that constant.*

Proof. First we assume that $\lim \frac{N_{n+1}}{N_n} = \lambda$ then as $n \to \infty$:

$$\lim \frac{N_{n+k}}{N_n} = \lim \left[\frac{N_{n+k}}{N_{n+k-1}} \times \frac{N_{n+k-1}}{N_{n+k-2}} \times \cdots \times \frac{N_{n+1}}{N_n}\right] = \lambda \times \lambda \cdots \times \lambda = \lambda^k \tag{3.101}$$

From Recursive Relation (3.89), we have as $n \to \infty$

$$\lim (N_{n+3}) = \lim (N_{n+2} + N_n). \tag{3.102}$$

That is:

$$\lim \frac{N_{n+3}}{N_n} = \lim \frac{N_{n+2}}{N_n} + 1. \tag{3.103}$$

Now using (3.101), we have $\lambda^3 = \lambda^2 + 1$ leading to $\lambda^3 - \lambda^2 - 1 = 0$. But this is precisely equation (3.90), whose positive root we found in to be approximately 1.46557. Thus $\lambda = 1.46557\ldots$ and this is the ratio of the consecutive terms of the Narayana's cows sequence. We call the limiting ratio $1.46557\ldots$ the Supergolden Ratio in view of the extension of the Golden Ratio obtained as the limiting ratio of two consecutive terms of the Fibonacci sequence. □

In addition, the Supergolden Ratio is the fourth smallest Pisot number OEIS: A092526).

Applications to cryptography

The principal applications of Narayana numbers in cryptography are as follows:

- **Foundations**
  - The Narayana universal code is generated using the Narayana sequence and constraining rules to ensure unique coding [KK16]. The number of bits required for the representation of the codeword follows the Narayana sequence. The code can be used to encode any positive integer, and decoding involves assigning the remaining bits with values from the Narayana sequence. A variant of Narayana coding can also be derived using a second-order Narayana sequence [DS19], with the third-order variant being even more beneficial for cryptographic applications [Çe21].

- **Implementation**
  - The Narayana sequence modulo p is shown to have good autocorrelation and crosscorrelation properties [Kir15]. It has good randomness properties, so they might be used for cryptographic and key distribution applications.

Further details

Remark on Perfect Secrecy. Perfect ciphers hold significant allure within the field of cryptography, having been widely implemented since C. Shannon's seminal publication, in which he delineated the characteristics of such ciphers and demonstrated the theoretical perfection of the one-time pad, also known as the Vernam cipher [Sha49].

The studied concept pertains to symmetric-key cryptography involving three entities: Alice, Bob, and Eve. Alice seeks to transmit a confidential message to Bob while ensuring its secrecy from Eve, who possesses the capability to intercept all communications between Alice and Bob. To achieve this, Alice and Bob employ a cipher with a secret key k (i.e., a sequence from a defined alphabet), which is pre-shared between them but remains unknown to Eve. When Alice intends to dispatch a message m, she initially encrypts m using the key k and subsequently transmits the encrypted message to Bob, who then decrypts it with the same key k. Eve intercepts the encrypted message and attempts decryption without knowledge of the key. The cryptosystem is deemed perfectly secure if Eve, even with unbounded computational resources and infinite time, is incapable of extracting any information regarding the original message. C. Shannon not only provided a rigorous definition of perfect (or unconditional) secrecy but also demonstrated that the one-time pad (or Vernam cipher) embodies such a secure system. A salient characteristic of this cryptosystem is that the length of the secret key must match the length of the message (or its entropy). This requirement often constrains practical implementation, as contemporary telecommunication systems routinely process and store vast quantities of data.

A fundamental characteristic of this cryptographic system is the requirement that the secret key be of equivalent length to the message or its entropy. This attribute is frequently impractical for many contemporary telecommunication systems that routinely transmit and store data on the scale of megabytes. Consequently, a logical approach involves the pre-encryption compression of messages using lossless data compression techniques, thus reducing the message length and, correspondingly, the length of the secret key, prior to applying the one-time pad [SP21; Rya23]. Furthermore, the expected length of a secret key can be approximated near the theoretical limit (i.e., Shannon entropy), contingent upon whether the probability distribution of the encrypted messages is known or unknown. In scenarios where the probability distribution is known, the well-established Huffman coding algorithm is applicable; conversely, when the distribution is unknown, a universal coding scheme (or standard compression algorithm) is employed.

In-depth analysis of sequence application. Integer number sequences have often been used to designate universal codes useful for compression activities employed in cryptography and described previously.

Universal coding of integers is a variable-length code for discrete memory-less sources with infinite alphabets, and the probability distribution of the sources does not require prior knowledge. The fundamental framework for universal coding of integers considers discrete memoryless sources $S = (P, \mathcal{A})$ with a countable alphabet set $\mathcal{A} = \mathbb{N}^+ = \{1, 2, 3, \cdots\}$ and a Decreasing Probability Distribution (DPD) P of $\mathbb{N}^+$(i.e., $\sum_{n=1}^{\infty} P(n) = 1$, and $P(m) \geq P(m+1) \geq 0$, for all $m \in \mathbb{N}^+$) [Eli75]. Let $H(P) = -\sum_{n=1}^{\infty} P(n) \log_2 P(n)$ denote the Shannon entropy of P. Let $\mathcal{C}$ be a variable-length code for the source $S = (P, \mathbb{N}^+)$; it maps the positive integers $\mathbb{N}^+$ onto the binary codewords $\{0, 1\}^*$. Let $L_{\mathcal{C}}(\cdot)$ denote the length function such that $L_{\mathcal{C}}(m) = |\mathcal{C}(m)|$, for all $m \in \mathbb{N}^+$, where $\mathcal{C}(m)$ is the corresponding codeword of m. Furthermore, $E_P(L_{\mathcal{C}}) = \sum_{n=1}^{\infty} P(n) L_{\mathcal{C}}(n)$ denotes the expected codeword length of $\mathcal{C}$. We say that $\mathcal{C}$ is universal if:

$$\frac{E_P(L_{\mathcal{C}})}{\max\{1, H(P)\}} \leq K_{\mathcal{C}} \tag{3.104}$$

for all DPDs P with $H(P) < \infty . K_{\mathcal{C}}$ is called the expansion factor of universal coding of integers $\mathcal{C}$, and $K_{\mathcal{C}}^* \triangleq \inf\{K_{\mathcal{C}} \mid \forall \, \text{DPD} \, P \text{ and } H(P) < \infty\}$ is called the minimum expansion factor of universal coding of integers $\mathcal{C}$. Moreover, $\mathcal{C}$ is called asymptotically optimal if $\mathcal{C}$ is universal and there exists a function $R_{\mathcal{C}}(\cdot)$ such that:

$$\frac{E_P(L_{\mathcal{C}})}{\max\{1, H(P)\}} \leq R_{\mathcal{C}}(H(P)) \tag{3.105}$$

for all DPDs P with $H(P) < \infty$ and:

$$\lim_{H(P) \to +\infty} R_{\mathcal{C}}(H(P)) = 1 \tag{3.106}$$

Let $L(j)$ denote the length of the j^{th} codeword c_j of the set of codewords C. Then:

- if, for all $j, L(j) \geq j^t$, for some constant $t > 0$, then the set is not universal;
- if, for all $j, L(j) \leq K_1 + K_2 \log_2 j$, for some constants K_1 and K_2, then the set is universal;
- if, for all $j, L(j) \geq K_1 + K_2 \log_2 j$, for some constant K_1 and K_2, where $K_2 > 1$, then the set is not asymptotically optimal.

All universal encoding methods for countable sources are constructive and do roughly the same thing. Two techniques can be distinguished: the message length strategy and the flag strategy [Cap90]. The length strategy focusses on encoding an integer by first encoding the length of the integer (i.e. binary), followed by the encoding of the integer itself. This method allows for a variable-length code that can efficiently represent small integers with shorter *bit* sequences while using longer sequences for larger integers. Unlike flag strategy, it is more about adding a specific pattern or prefix (flag) to the code to indicate the range or category of the integer and then encoding the integer within that context. This method is often used in schemes where integers are categorised into different ranges and each range has a unique prefix or flag that helps decode.

(**Fibonacci codes**). The Fibonacci family of codes is probably the most famous flag pattern strategy for universal coding of integers [AF87]. The Fibonacci code is closely related to the Zeckendorf representation, a positional numeral system that uses Zeckendorf's theorem and has the property that no number has a representation with consecutive 1's. The Fibonacci code word for a particular integer is exactly the integer's Zeckendorf representation with the order of its digits reversed and an additional "1" appended to the end.

Theorem 3.1.4. *(Zeckendorf's theorem). Every positive integer has a unique representation as the sum of non consecutive Fibonacci numbers.*
Let n be a positive integer. Then there is a unique increasing sequence $(c_i)_{i=0}^{k}$ such that $c_i \geq 2$ and $c_{i+1} > c_i + 1$ for $i \geq 0$, and that:

$$n = \sum_{i=0}^{k} F_{c_i} \tag{3.107}$$

We will call such a sum the Zeckendorf representation for n.

Proof. Zeckendorf's theorem has thus two parts [Zec72]:

1. (*Existence*). Every positive integer n has a Zeckendorf representation.
   We see that $1 = F_2, 2 = F_3, 3 = F_4$, and $4 = F_2 + F_4 = 1 + 3$. Suppose now that we can find such a representation for all positive integers up to k. If $k+1$ is a Fibonacci number, then that provides the Zeckendorf representation. If $k+1$ is not a Fibonacci number, then $\exists j \in \mathbb{N}$ is such that $F_j < k+1 < F_{j+1}$. Define $a = k+1-F_j$, so $a \leq k$, meaning a has a Zeckendorf representation by hypothesis. We also note that:

$$\begin{gathered} F_j + a = k+1 < F_{j+1} = F_j + F_{j-1} \\ a < F_{j-1}. \end{gathered} \tag{3.108}$$

   Thus, the Zeckendorf representation of a does not contain a F_{j-1} term, so $k+1 = F_j + a$ will yield a Zeckendorf representation for $k+1$. This proves the existence of the Zeckendorf representation for positive integers by induction.

2. (*Uniqueness*). No positive integer n has two different Zeckendorf representations.
   Let n be a positive integer with two non-empty sets of terms S and T that form Zeckendorf representations of n. Let $S' = S \backslash T$ and $T' = T \backslash S$. Since both sets lost the same common elements, we still have:

$$\begin{gathered} \sum_{x \in S} x - \sum_{a \in S \cap T} a = \sum_{y \in T} y - \sum_{b \in S \cap T} b \\ \sum_{x \in S'} x = \sum_{y \in T'} y \end{gathered} \tag{3.109}$$

   Thus, if S' or T' is empty, it will produce a sum of 0. Since all terms are non-negative, the other sum, equal to 0, must also be empty, which means that $S' = T' = \varnothing$, so $S = S \cap T = T$. Let us now assume that both sets S'

and T' are non-empty. Let $F_s = \max S'$ and $F_t = \max T'$. Since $S' \neq T'$, we may say without loss of generality that $F_s < F_t$. Thus, we may say that:

$$\sum_{x \in S'} x < F_{s+1} \leq F_t. \tag{3.110}$$

Since the sums over S' and T' are non-negative and equal, this is a contradiction, so $S' = T' = \varnothing$, and $S = T$.

Definition 3.1.11. *(Fibonacci codes). For a number N, if $d(0), d(1), \ldots, d(k-1), d(k)$ represent the digits of the code word representing N then we have:*

$$N = \sum_{i=0}^{k-1} d(i)F(i+2), \text{ and } d(k-1) = d(k) = 1 \tag{3.111}$$

where $F(i)$ is the i th Fibonacci number, and so $F(i+2)$ is the i th distinct Fibonacci number starting with $1, 2, 3, 5, 8, 13, \ldots$ The last bit $d(k)$ is always an appended bit of 1 and does not carry place value.

To encode an integer N:

1. Find the largest Fibonacci number equal to or less than N; subtract this number from N, keeping track of the remainder.
2. If the subtracted number was the ith Fibonacci number $F(i)$, put a 1 in place $i-2$ in the code word (counting the leftmost digit as place 0).
3. Repeat the previous steps, substituting the remainder for N, until a remainder of 0 is reached.
4. Place an additional 1 after the rightmost digit in the code word.

To decode a code word, remove the final "1", assign the remaining the values $1, 2, 3, 5, 8, 13\ldots$ (Fibonacci sequence - OEIS: A000045) to *bits* in the code word, and sum the values of "1" *bits*.

Fibonacci coding possesses a distinctive attribute that renders it appealing in comparison to other universal codes: it exemplifies a self-synchronizing code, which facilitates the recovery of data from a compromised stream. With most other universal codes, if a single *bit* is altered, none of the data that follows it will be correctly read.

(**Narayana codes**). To generate Narayana code as a generalisation of Fibonacci universal code, we need to be able to map any given positive integer representing source code into variable-length code word in a manner used earlier [Tho07].

A more general Narayana sequence $N_a(k)$ is given by $a, b, c, a+c, a+b+c, a+b+2c, 2a+b+2c, 3a+2b+4c$ and so on with $a = 1, b = 2$ and $c = 3$.

Definition 3.1.12. *A variant of the Narayana coding scheme can be obtained by defining the second-order variant Narayana sequence, $VN_a^{(2)}(k)$, such that $b = 3 - a$ and $c = 1 - a$ [KK16]. This yields:*

$$\begin{cases} VN_a^{(2)}(0) = a \quad (a \in Z) \\ VN_a^{(2)}(1) = 3 - a \\ VN_a^{(2)}(2) = 1 - a \\ VN_a^{(2)}(k) = VN_a^{(2)}(k-1) + VN_a^{(2)}(k-3) \quad \text{for} \quad k \geq 3 \end{cases} \tag{3.112}$$

In the light of the above definition, we get a variant of the Narayana series:

$$VN_{-2}^{(2)}(n) = \{-2, 5, 3, 1, 6, 9, 10, 16, 25, \ldots\} \tag{3.113}$$

and

$$VN_{-5}^{(2)}(n) = \{-5, 8, 6, 1, 9, 15, 16, 25, 40, \ldots\}, \text{ and so on.} \tag{3.114}$$

Moreover, we obtain there is no Zeckendorf representation for integers 3 and 15 using the sequence $VN_{-1}^{(2)}(k) = \{-1, 4, 2, 1, 5, 7, 8, 13\}$, and integers 2 and 13 can't be represented using sequence $VN_{-3}^{(2)}(k) = \{-3, 6, 4, 1, 7, 11, 12, 19\}$. Upon examination of the second order variant Narayana codes [DS19] we obtained:

- for the only positive integer $k = 1$, the second order variant Narayana code $VN_u^{(2)}(k)$ exactly exists for $u = -1, -2, \ldots, -20$;
- for $1 \le k \le 50$, there is at most j consecutive undetectable values (NA) the second order variant of Narayana code in $VN_{-j}^{(2)}(k)$ column in which $1 \le j \le 20$;
- as long as j raises, the detectable of Narayana code is reduced in $VN_{-j}^{(2)}(k)$ column in which $1 \le j \le 20$.

Definition 3.1.13. *The third order variant Narayana sequences $VN_a^{(3)}(k)$ is described with the sequence $\{a, b, c, a+c, a+b+c, 2a+b+2c, 3a+2b+3c, \ldots\}$ where $b = 3 - a$ and $c = 1 - a$, that is:*

$$\begin{cases} VN_a^{(3)}(1) = a \\ VN_a^{(3)}(2) = 3 - a \\ VN_a^{(3)}(3) = 1 - a \\ VN_a^{(3)}(4) = 1 \\ VN_a^{(3)}(5) = 4 - a \\ VN_a^{(3)}(k) = VN_a^{(3)}(k-1) + VN_a^{(3)}(k-3) + VN_a^{(3)}(k-5) \quad \textit{for} \quad k \ge 6 \end{cases} \tag{3.115}$$

The third order variant Narayana coding scheme can be obtained by defining the third order variant Narayana sequence, $VN_u^{(3)}(k)$, such that $b = 3 - a$ and $c = 1 - a$ [Çe21]. In variant Narayana sequences, some integers have more than one Narayana code, while others have no Narayana code. For example, for the second-order variant Narayana sequence $VN_{-1}^{(2)}(k) = \{-1, 4, 2, 1, 5, 7, 8, 13\}$ while there is no Narayana code for integers $k = 2$ and $k = 11$ [DS19], there are two Narayana codes for integer $k = 4$. These codes are 011 and 100011. Similarly, for the third-order variant Narayana sequence $VN_{-7}^{(3)}(k) = \{-7, 10, 8, 1, 11, 12, 23, 42\}$ while there is no Narayana code for integers $k = 7$ and $k = 40$, there are two Narayana codes for integer $k = 1$. These codes are 000111 and 10111. We obtain the third order variant of Narayana codes $VN_u^{(3)}(k)$ or undetectable values (NA) of the positive integer k for $1 \le k \le 50$ and for $u = -1, -2, \ldots, -20$. The results for the third order variant Narayana codes are:

- for the positive integers $k = 1, 2, 3, 4, 5, 6$, the third order variant of Narayana code $VN_u^{(3)}(k)$ exactly exists for $u = -1, -2, \ldots, -20$.

– for $1 \leq k \leq 50$, there are at most j consecutive undetectable values (NA) the third-order variant of the Narayana code in $VN^{(3)}_{-(6+j)}(k)$ column in which $1 \leq j \leq 14$.

– for $1 \leq k \leq 50$, as long as j increases, the detectable third-order variant Narayana code is reduced in $VN^{(3)}_{-(6+j)}(k)$ column in which $1 \leq j \leq 14$.

3.1.8 Other sequences

Discussion of other sequences that have not had cryptographic applications as widespread in the literature as the previous ones, but which are significant.

Fermat primes

Definition 3.1.14. *A Fermat prime is a Fermat number $2^{2^n}+1$, that is prime.*

Fermat conjectured in 1650 that every Fermat number is prime, and Eisenstein in 1844 proposed as a problem the proof that there are an infinite number of Fermat primes [Rib96]. However, currently, the only Fermat numbers F_n for $n \geq 5$ for which primality or compositeness has been established are all composite.
OEIS. The first terms of the sequence are available in the OEIS database:

A-number	A019434
Name	Fermat primes: primes of the form $2^{2^k}+1$, for some $k >= 0$.
Data	$3, 5, 17, 257, 65537, \cdots$
Offset	$1, 1$
Link	https://oeis.org/A019434

Table 3.9 Fermat primes in OEIS database.

Applications to cryptography. Historically, the use of Fermat numbers in cryptography is primarily related to their role in the generation and testing of large prime numbers, which are fundamental to many cryptographic protocols and systems. Recently, Fermat numbers have been proposed as the main module for system computations in Fully Homomorphic Encryption (FHE) cryptosystems [Jou19].

Proth primes

Definition 3.1.15. *A Proth number is a natural number of the form $k \cdot 2^s + 1$ where $k, s \in \mathbb{N}, k$ is odd and $k < 2^s$. A Proth prime is a Proth number that is prime in $\mathbb{N}$.*

Fermat numbers $(k = 1, s = 2^t)$ and Cullen numbers $(k = s)$ are special cases of Proth numbers.

Their name was given in honour of the French mathematician François Proth who introduced these numbers in his 1878 paper [Pro78]. He developed a theorem to determine whether a number of the form $k \cdot 2^n + 1$ is prime or composite. This theorem is based on the evaluation of $k \cdot 2^n \pmod{n}$, where n must be greater than k. If the result satisfies specific criteria, the number is considered a Proth prime.

To date, the largest Proth prime discovered is $10223 \cdot 2^{31172165} + 1$ having 9383761 decimal digits, found by the PrimeGrid distributed computing project in 2016. This is the largest known non-Mersenne prime number.

It is also noted that the theory of Proth numbers and Proth primes is surprisingly limited, and even basic results were not documented in the scientific literature.
OEIS. The first terms of the sequence are available in the OEIS database:

A-number	A080076
Name	Proth primes: primes of the form $k \cdot 2^m + 1$ with odd $k < 2^m$, $m \geq 1$.
Data	$3, 5, 13, 17, 41, 97, 113, 193, 241, 257, 353, 449, 577, 641, 673, 769, \cdots$
Offset	1, 1
Link	https://oeis.org/A080076

Table 3.10 Proth primes in OEIS database.

Applications to cryptography. Proth primes have applications in cryptography; for example, they can optimise the Boer reduction between the Diffie–Hellman problem and the Discrete logarithm problem [Bro14]. The prime number $55 \cdot 2^{286} + 1$ has been used in this way. Furthermore, Proth primes are used to design efficient and flexible Number Theoretic Transform (NTT) architectures for Post-Quantum Cryptography (PQC) and FHE [PS21].

Gaussian integers

Definition 3.1.16. *The Gaussian integers are the set $\mathbb{Z}[i] = \{x + iy : x, y \in \mathbb{Z}\}$ of complex numbers whose real and imaginary parts are both integers.*

$\mathbb{Z}[i]$ is a ring (really a subring of $\mathbb{C}$) since it is closed under addition and multiplication:

$$(x + iy) + (p + iq) = (x + p) + i(y + q) \tag{3.116}$$

$$(x + iy)(p + iq) = (xp - yq) + i(xq + yp) \tag{3.117}$$

The second follows from the fact that i satisfies the quadratic polynomial $i^2 + 1 = 0$. The Gaussian integers have many special properties that are similar to those of the integers, and they are named after the German mathematician Carl Friedrich Gauss.
Gaussian primes. As Gaussian integers form a principal ideal domain, they also form a unique factorisation domain. This implies that a Gaussian integer is irreducible (that is, it is not the product of two non-units) if and only if it is prime (that is, it generates a prime ideal).
The prime elements of $\mathbb{Z}[i]$ are also known as Gaussian primes. An associate of a Gaussian prime is also a Gaussian prime. The conjugate of a Gaussian prime is also a Gaussian prime (this implies that Gaussian primes are symmetric about the real and imaginary axes).
A positive integer is a Gaussian prime if and only if it is a prime number that is congruent to 3 modulo 4 (that is, it may be written $4n + 3$, with n a non-negative integer). The other prime numbers are not Gaussian primes, but each is the product of two conjugate Gaussian primes.

A-number	A002145
Name	Primes of the form $4k+3$ (natural primes which are also Gaussian primes).
Data	$3, 7, 11, 19, 23, 31, 43, 47, 59, 67, 71, 79, 83, 103, 107, 127, 131, \cdots$
Offset	$1, 1$
Link	`https://oeis.org/A002145`

Table 3.11 Gaussian primes in OEIS database.

OEIS. The first terms of the sequence are available in the OEIS database:
Applications to cryptography. Gaussian integers are used to extend the traditional RSA algorithm within its domain [EES02]. The extension also involves adapting existing algorithms, such as the extended Euclidean algorithm, to the domain of Gaussian integers. This allows for the definition of a modified version of Euler's totient function, necessary for choosing the encryption and decryption exponents in RSA, and for understanding how to encrypt and decrypt messages within this new context. The method is demonstrated to represent messages as Gaussian numbers, and the product of two Gaussian primes is used to form the modulus of the cryptographic system, a key component of the security of RSA. Additionally, a new hybrid public/private key cryptography scheme uses Perfect Gaussian Integer Sequences (PGIS) with a period of $N = pq$, where p and q are odd primes [HLCX21]. This novel scheme is based on circular convolution over PGISs, which is shown to be a trapdoor one-way permutation function, allowing for both cipher encryption and digital signatures. The authors assert that the security level of this new scheme is comparable to that of the RSA system, while potentially offering better performance in terms of system capacity due to the abundance of PGISs available. Additionally, the proposed scheme is especially suitable for use on the Internet of Things (IoT) platforms, where lightweight cryptographic functions are needed due to the limited resources of IoT devices.

3.2 Our considerations

A comprehensive review of the literature reveals that integer sequences have exerted a profound influence in diverse domains of cryptography. As mathematicians engage in more rigorous exploration of integer sequences, they discern novel patterns and interconnections that bear significant practical implications within these domains. However, it is crucial to emphasise that certain sequences exhibit a higher degree of relevance with respect to their applicability to specific subdomains of cryptography. Nevertheless, it is evident that all principal areas of cryptography examined in our study have been affected by at least one integer sequence:

This elucidates the extensive scope and efficacy of employing such sequences over time. Specifically, a review of the literature underscores that integer sequences have been effectively used in public-key cryptography, where they have demonstrated robust security properties. Furthermore, their application in symmetric key cryptography underscores their versatility and appropriateness for various cryptographic algorithms, thereby underpinning the development of resilient and secure crypto-

Macro-areas	Prime	Mersenne	Sophie German	Fibonacci	Lucas	Catalan	Narayana
Foundations		■	■	■	■		■
Cryptographic hash functions	■	■		■			
Secret-key cryptography			■			■	
Public-key cryptography	■	■	■		■	■	
Cryptographic protocols	■		■			■	
Implementation	■				■	■	■
Attacks and cryptanalysis					■		
Steganography				■		■	

Table 3.12 Using the main integers sequences in the principal macro-areas of the cryptography domain.

graphic systems. The diverse applications of integer sequences thus underscore their flexibility and adaptability in different cryptographic contexts. Our analysis suggests that it is imperative to continue investigating and harnessing the potential of integer sequences within the domain of cryptography, given their multifaceted implications and the broad spectrum of contexts in which they can be effectively employed.

Chapter 4

A new generalized family of Integer Sequences

4.1 Sieve theory

A sieve represents a method designed to restrict the size of a set by removing elements that exhibit "undesirable properties," typically related to number theory. Such undesirable traits may include divisibility by a prime number from a specified set, other multiplicative limitations (e.g., divisibility by a perfect square), or belonging to a specific set of residue classes. The inclusion-exclusion principle provides an exact solution; however, it results in 2^k terms for k properties, making practical computations challenging. A sieve serves as an approach to approximate the count of "desirable" elements within the set utilising the $k^{O(1)}$ terms. Although not precise, sieves frequently achieve very accurate estimates of the set's size.

The original method for identifying primes is the Sieve of Eratosthenes, a well-known technique for constructing a table of prime numbers by systematically eliminating integers that are divisible by smaller primes while retaining the primes themselves. In the period 1915-1922, Viggo Brun developed a modern sieve to address renowned unsolved problems like Goldbach's Conjecture and the Twin Prime problem, although these efforts have not yet met with success [Bru15]. Sieve methods have since become widely utilized in number theory and serve as valuable tools in tackling various other challenges, such as investigating Diophantine equations.

4.1.1 The Sieve problem

Let $\mathcal{A}$ be a finite set of objects and let $\mathcal{P}$ be an index set of primes such that to each $p \in \mathcal{P}$ we have associated a subset $\mathcal{A}_p$ of $\mathcal{A}$. The sieve problem is to estimate, from above and below, the size of the set:

$$\mathcal{S}(\mathcal{A}, \mathcal{P}) := \mathcal{A} \setminus \cup_{p \in \mathcal{P}} \mathcal{A}_p \tag{4.1}$$

This is the formulation of the problem in the most general context. Of course, the "explicit" answer is given by the familiar inclusion-exclusion principle in combinatorics. More precisely, for each subset I of $\mathcal{P}$, denote by:

$$\mathcal{A}_I := \cap_{p \in I} \mathcal{A}_p \tag{4.2}$$

Then the inclusion-exclusion principle gives us:

$$\#\mathcal{S}(\mathcal{A},\mathcal{P}) = \sum_{I\subseteq\mathcal{P}} (-1)^{\#I}\#\mathcal{A}_I, \tag{4.3}$$

where for the empty set $\emptyset$ we interpret $\mathcal{A}_\emptyset$ as $\mathcal{A}$ itself. This formula is the basis in many questions of probability theory (see the exercises).

In number theory we often take $\mathcal{A}$ to be a finite set of positive integers and $\mathcal{A}_p$ to be the subset of $\mathcal{A}$ consisting of elements lying in certain specified congruence classes modulo p. For example, if $\mathcal{A}$ is the set of natural numbers $\leq x$ and $\mathcal{A}_p$ is the set of numbers in $\mathcal{A}$ divisible by p, then the size of $\mathcal{S}(\mathcal{A},\mathcal{P})$ will be the number of positive integers $n \leq x$ coprime to all the elements of $\mathcal{P}$. Estimation $\#\mathcal{S}(\mathcal{A},\mathcal{P})$ is a fundamental question that arises in many disguises in mathematics and forms the focus of attention of all sieve techniques. We will illustrate this in later chapters.

We could also reverse the perspective. That is, we can think of $\mathcal{S} = \mathcal{S}(\mathcal{A},\mathcal{P})$ as a given set whose size we want to estimate. We seek to do this by looking at its image modulo primes $p \in \mathcal{P}$ for some set of primes $\mathcal{P}$.

The Sieve of Eratosthenes

The original sieve is, of course, the sieve of Eratosthenes for finding prime numbers. To find the prime numbers in $\{2,\dots,n\}$, you repeat the following operation as long as there are unmarked numbers: find the first unmarked number p, mark it as prime, then mark $2p, 3p, \dots$ as composite until you get to a number greater than n.

Of course, one need only sift out multiples of primes up to $n^{1/2}$ in order to leave only primes behind. More generally, if one is only able to sift out multiples of primes up to n^α, what remain are numbers with no prime factors less than n^α. In particular, any such number has at most $\lfloor \alpha^{-1} \rfloor$ prime factors, and so is in some sense "nearly prime".

Of course, in the process of sieving, many numbers will be sifted out more than once. If one wants to draw any sort of quantitative conclusion from this process, one must keep track of the multiple counting; this suggests using inclusion-exclusion.

4.1.2 The principle of inclusion-exclusion

Let S be a finite set, and let $P_1,\dots,P_n$ be subsets of S. Think of each P_i as containing the elements of S with a certain property.

Suppose that we have some way to count the number of elements in the intersection of any subcollection of P_i, but what we really want is to count the complement of the union of all of the P_i. The formula that computes this is:

$$\#\left(S\backslash\left(P_1\cup\cdots\cup P_n\right)\right) = \sum_{T\subseteq\{1,\dots,n\}} (-1)^{\#T}\#\left(\bigcap_{t\in T} P_t\right). \tag{4.4}$$

Proof. if $s \in S$ belongs to m of the subsets, then the number of times it gets counted on the right side is:

$$\binom{m}{0} - \binom{m}{1} + \cdots \tag{4.5}$$

which equals 1 if $m = 0$ and vanishes otherwise. More generally, if $f : S \to \mathbb{C}$ is some function, and we want to compute the sum of f over the complement of the P_i, we have:

$$\sum_{s \in S \setminus (P_1 \cup \cdots \cup P_n)} f(s) = \sum_{T \subseteq \{1,\ldots,n\}} (-1)^{\#T} \left(\sum_{s \in \cap_{t \in T} P_t} f(s) \right). \tag{4.6}$$

In number theory, we are often taking $S = \{1, \ldots, N\}$ and taking the sets $P_1, P_2, \ldots$ to be the sets of multiples of certain small primes. It is convenient to rewrite the principle of inclusion-exclusion in terms of the arithmetic function μ, the Möbius function:

$$\mu(n) = \begin{cases} (-1)^d & n = p_1 \cdots p_d \quad (p_1, \ldots, p_d \text{ distinct, } d \geq 0) \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

□

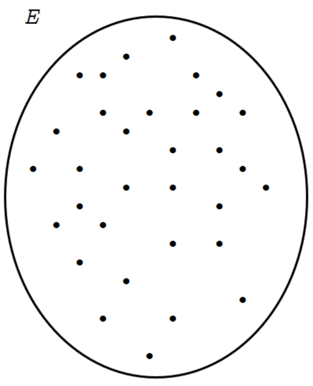
Additional perspectives on Sieve theory can be found in [GHH97; CM05; Gre13]

4.2 Our contribution

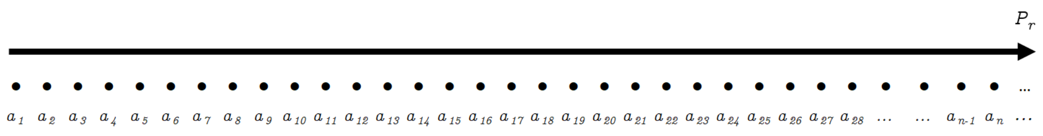
4.2.1 New model for sub-sequence generation

Inspired by the foundational principle of the previously discussed sieve models, we have devised an innovative algorithmic approach for selecting elements from a set to form a sub-sequence.

Let E be an unordered and non-empty set of elements (•). We assume that there exists a property P_r that allows to order the elements of E and to attribute an index n to each that we define as a_n.



With regard to the property P_r, the set formed as $E = \{a_1, a_2, a_3, \ldots, a_{n-1}, a_n\}$ is capable of being represented graphically in the following manner:

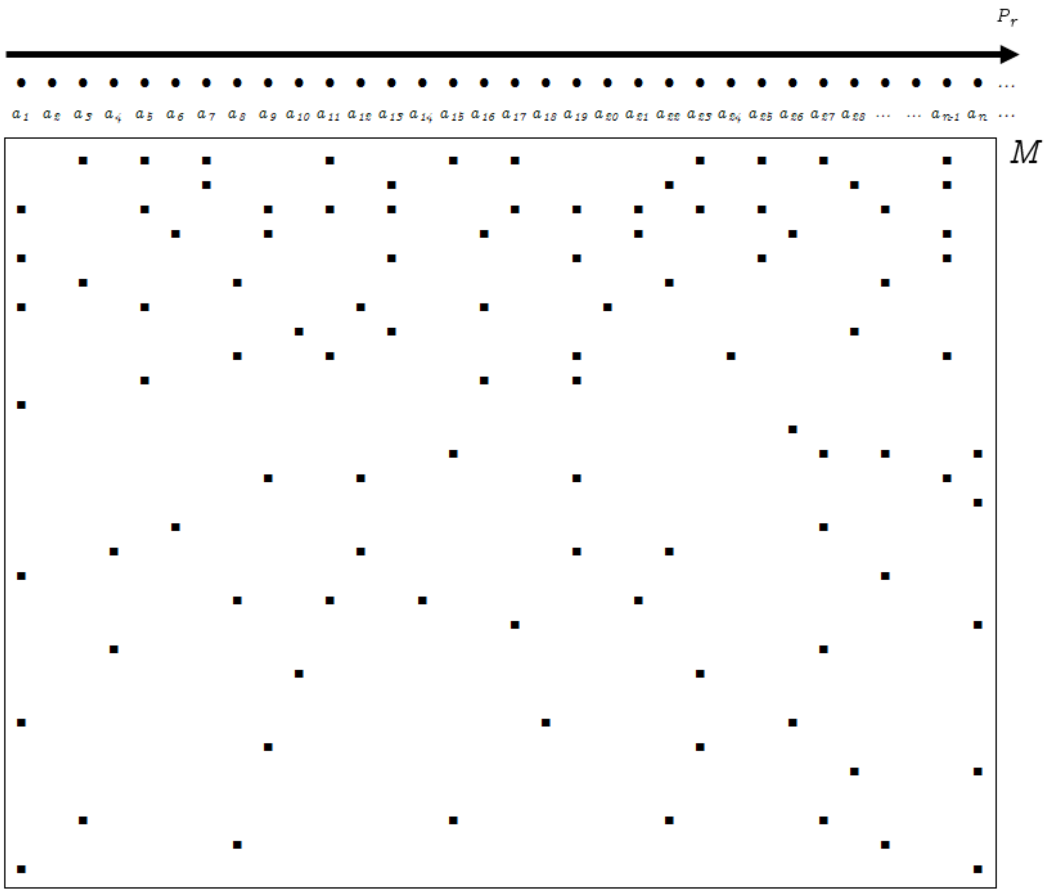

P_r
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12} a_{13} a_{14} a_{15} a_{16} a_{17} a_{18} a_{19} a_{20} a_{21} a_{22} a_{23} a_{24} a_{25} a_{26} a_{27} a_{28} … … a_{n-1} a_n …

Definition 4.2.1. *(**Model**). A model $\mathcal{M}_{IS}$ that identifies a family of integer sequences is a triplet of algorithms (Pattern, Shift, Sieve) that takes an input sequence F_n and returns a sub-sequence S_m of a_n.*

In essence, all sequences S_m that can be generated by the same model $\mathcal{M}_{IS}$*(Pattern, Shift, Sieve)* corresponding to a specific F_n are part of the same family of integer sequences.

Let us now give a definition of the three algorithms that make up our model $\mathcal{M}_{IS}$*(Pattern, Shift, Sieve)*.

Definition 4.2.2. *(**Pattern algorithm**). An algorithm (M) that assigns a pattern (■) to the sequence elements a_n of the set E.*

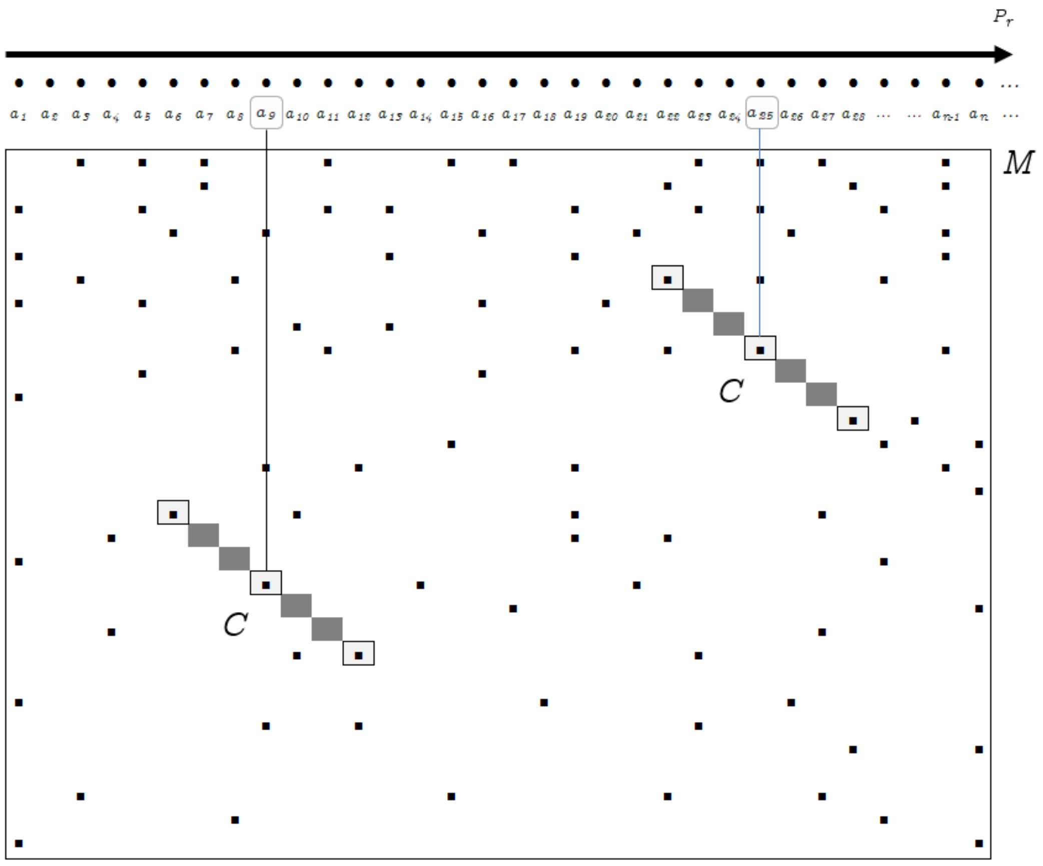

P_r
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12} a_{13} a_{14} a_{15} a_{16} a_{17} a_{18} a_{19} a_{20} a_{21} a_{22} a_{23} a_{24} a_{25} a_{26} a_{27} a_{28} ... ... a_{n-1} a_n ...
M

Definition 4.2.3. *(**Sieve algorithm**). An algorithm (C) that extracts from a_n a sub-sequence S_m whose elements satisfy specific conditions in the patterns (■) assigned to the elements a_n of the set E:*

$$S_m = C[M]. \tag{4.8}$$

In the following example, the algorithm C is designed to select the elements a_n of the set E that satisfy a specific condition on the patterns distributed by the algorithm M such that:

$$S_m = C[M] = \{a_9, a_{25}, ...\} \tag{4.9}$$



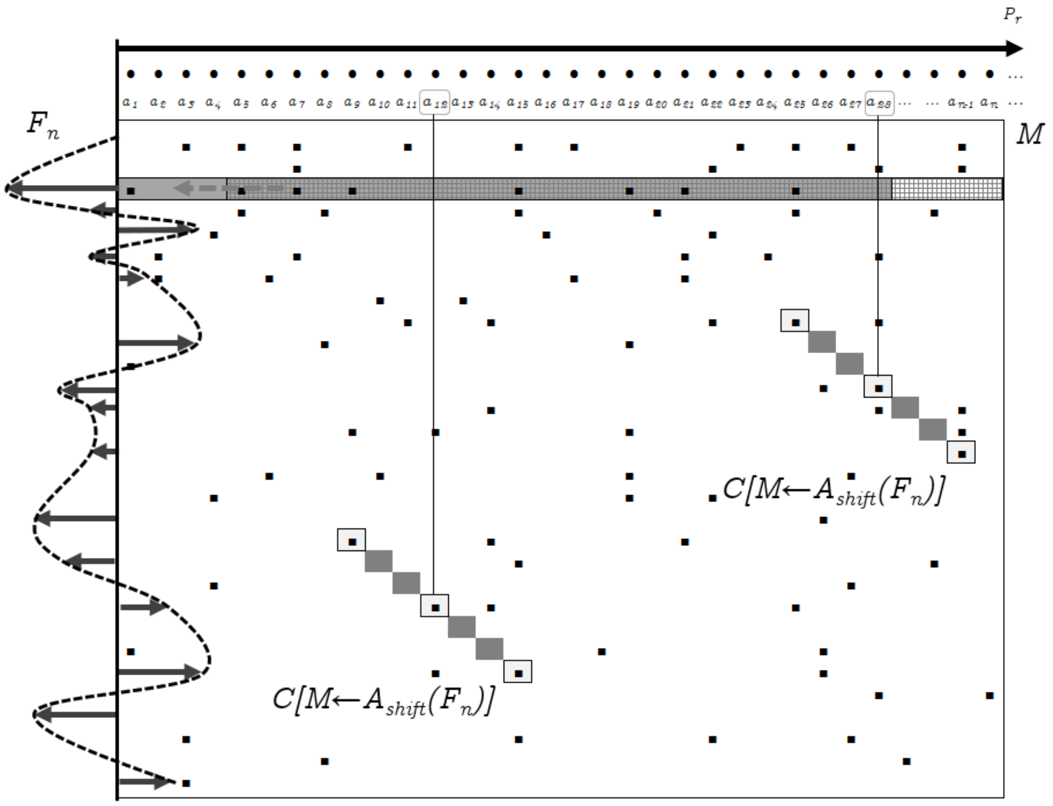
P_r
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12} a_{13} a_{14} a_{15} a_{16} a_{17} a_{18} a_{19} a_{20} a_{21} a_{22} a_{23} a_{24} a_{25} a_{26} a_{27} a_{28} … … a_{n-1} a_n …
M
C
C

Definition 4.2.4. ***(Shift algorithm)****. An algorithm* (A_{shift}) *that implements a translation to patterns* (■) *through a sequence* F_n*:*

$$S_m = C[M \leftarrow A_{shift}(F_n)]. \tag{4.10}$$

In the illustrated example, if we apply the Shift algorithm A_{shift} to the model M and then the Sieve algorithm C, we will obtain a new sub-sequence of a_n different from 4.9:

$$S_m = C[M \leftarrow A_{shift}(F_n)] = \{a_{12}, a_{28}, ...\} \tag{4.11}$$



P_r
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12} a_{13} a_{14} a_{15} a_{16} a_{17} a_{18} a_{19} a_{20} a_{21} a_{22} a_{23} a_{24} a_{25} a_{26} a_{27} a_{28} … … a_{n-1} a_n …
F_n
M
$C[M \leftarrow A_{shift}(F_n)]$
$C[M \leftarrow A_{shift}(F_n)]$

Considerations

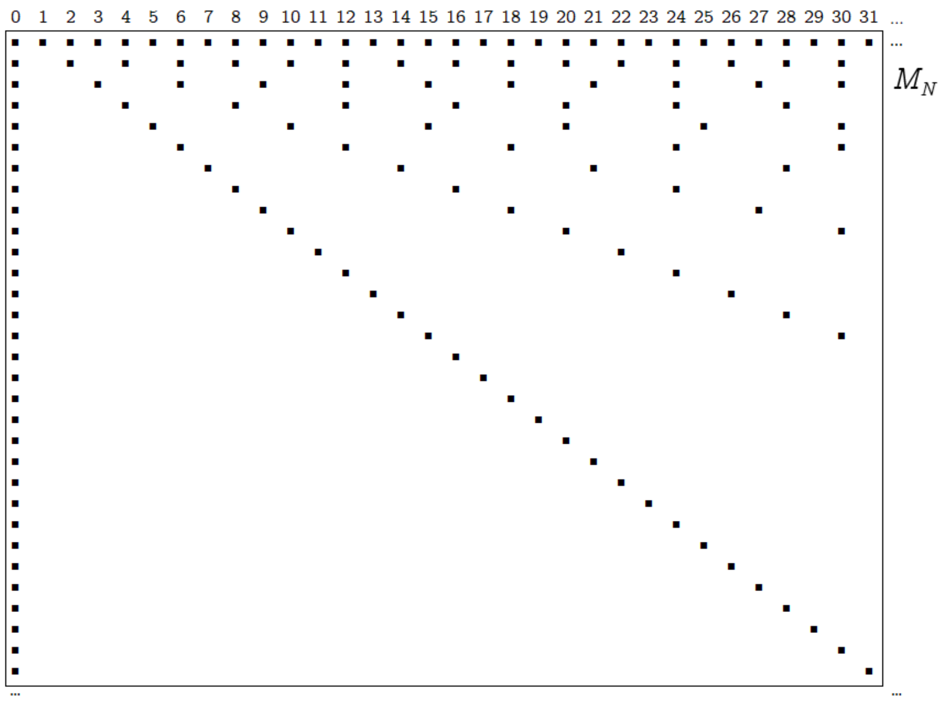
With regards to the modeling construct denoted by $\mathcal{M}_{IS}$ *(Pattern, Shift, Sieve)*, we are able to undertake initial preliminary analyses:

– The sequence S_m contains significantly fewer elements compared to the size of the model $\mathcal{M}_{IS}$ and the elements within F_n, where $m \ll n$.

– As specified, the algorithm $A_{shift}(F_n)$ generates different pattern distributions (■) for each F_n. This distribution characterises a "model state", represented as $ST_{[w]}$. The full range of potential distributions w that $\mathcal{M}_{IS}$ can adopt constitutes the set of model states $W = \{w_1, w_2, \cdots, w_g\}$.

4.2.2 Application of $\mathcal{M}_{IS}$ to $\mathbb{N}$

We will now use our sequence generation model $\mathcal{M}_{IS}$ applied to the sequence of natural numbers $\mathbb{N}$, with which to extract sub-sequences S_m of $\mathbb{N}$, which we will denote with $\mathcal{M}_{IS}^{\mathbb{N}}$. Specifically, we introduce the Pattern algorithm $M_{\mathbb{N}}$ (1), which allocates patterns related to each divisor number.



Algorithm 1: Pattern algorithm $M_{\mathbb{N}}$

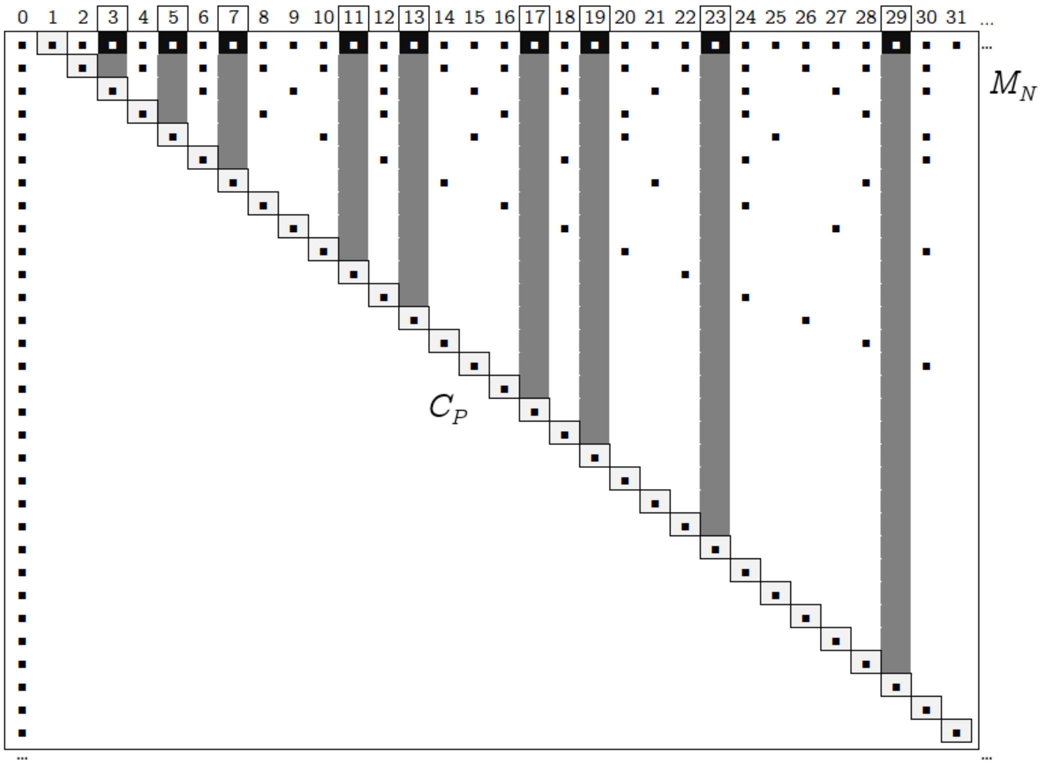
Input: dimension n of $M_{\mathbb{N}}$

Output: distribution of patterns (■)

```
for $n \leftarrow 0$ to *dimension* do
    if $n = 0$ then
        for $i \leftarrow 0$ to *dimension* do
            line$[n, i]$ = ■;
        end
    end
    else
        $j = n$;
        for $i \leftarrow 0$ to *dimension* do
            if $j < n$ then
                $j \leftarrow j + 1$;
                line$[n, i] \leftarrow$ null;
            end
            else
                line$[n, i]$ = ■;
                $j \leftarrow 0$;
            end
        end
    end
end
return *distribution array*
```

Let us now define in the following figure the Sieve algorithm $C_{\mathcal{P}}$ (2) corresponding to the selection criterion of prime numbers p over $\mathbb{N}$. Therefore, we will have that:

$$C_{\mathcal{P}}[M_{\mathbb{N}}] = \{p_1, p_2, \cdots, p_{n-1}, p_n\} \tag{4.12}$$



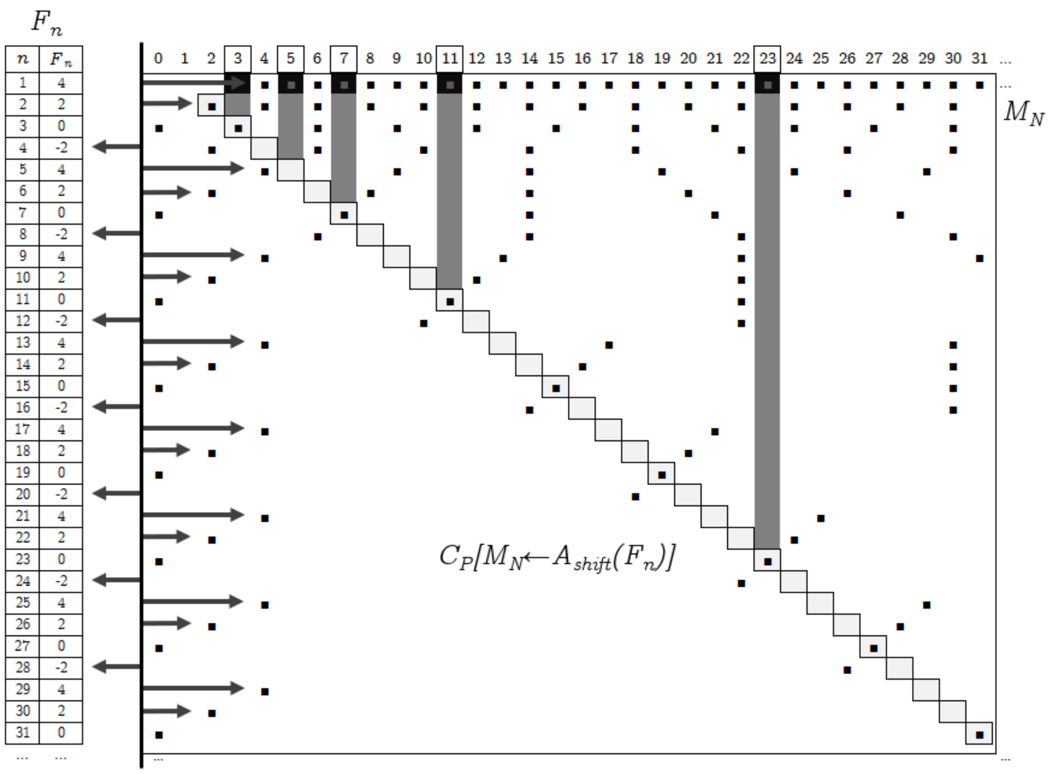
Algorithm 2: Sieve algorithm $C_{\mathcal{P}}$

```
Input: distribution of patterns (■)
Output: sequence $S_m$
1  for $i \leftarrow 3$ to $dimension$ do
2      $j \leftarrow 0$;
3      for $n \leftarrow 0$ to $i - 1$ do
4          if $line[n, i] = null$ then
5              $j \leftarrow j + 1$;
6              if $j = i - 2$ then
7                  $sequence \leftarrow i$;
8              end
9          end
10     end
11 end
12 return $sequence$
```

Example of $C_{\mathcal{P}}[M_{\mathbb{N}} \leftarrow A_{shift}(F_n)]$

Suppose now we apply a Shift algorithm A_{shift} (3) where the shift sequence F_n is defined as follows:

$$F_n = \{4, 2, 0, -2, 4, 2, 0, -2, 4, 2, 0, -2, 4, 2, 0, -2, 4, 2, 0, -2 \cdots\} \tag{4.13}$$



n	F_n
1	4
2	2
3	0
4	-2
5	4
6	2
7	0
8	-2
9	4
10	2
11	0
12	-2
13	4
14	2
15	0
16	-2
17	4
18	2
19	0
20	-2
21	4
22	2
23	0
24	-2
25	4
26	2
27	0
28	-2
29	4
30	2
31	0
...	...

We will therefore obtain a sub-sequence S_m extracted from $\mathbb{N}$:

$$C_{\mathcal{P}}[M_{\mathbb{N}} \leftarrow A_{shift}(F_n)] = \{3, 5, 7, 11, 23, 47, 71, 83, 107, 131, 167, 227, 311, 383, \cdots\} \tag{4.14}$$

As shown, the elements of the sequence S_m are all prime numbers, and such an original sequence is not present in the OEIS database!
In Appendix 6, several sequences S_m produced by the model $\mathcal{M}_{IS}$ are presented as illustrative, though not comprehensive, examples.

Considerations

Upon examining the model $\mathcal{M}_{IS}^{\mathbb{N}}$ as defined within $\mathbb{N}$, we can derive the following insights:

1. *line*[n,i], which represents the translation of a row in A_{shift}, can take on n different positions. Therefore, the set of model states W will contain $n!$ elements ($W = n!$).

2. For $ST_{[w_0]}$, corresponding to $F_n^{ST_{[w_0]}} = \{0, 0, 0, \cdots, 0\}$, we will have:

$$\mathcal{P} = C_{\mathcal{P}}[M_{\mathbb{N}} \leftarrow A_{shift}(F_n^{ST_{[w_0]}})] \tag{4.15}$$

Algorithm 3: Shift algorithm to Pattern algorithm $M_{\mathbb{N}} \leftarrow A_{shift}(F_n)$

```
Input: dimension $n$ of $M_{\mathbb{N}}$
       sequence $F_n$
Output: distribution of pattern (■)
for $n \leftarrow 0$ to dimension do
    if $n = 0$ then
        for $i \leftarrow 0$ to dimension do
            line$[n, i]$ = ■;
        end
    end
    else
        $m \leftarrow n + 1$;
        $j \leftarrow n - (F_n \bmod m)$;
        for $i \leftarrow 0$ to dimension do
            if $j < n$ then
                $j \leftarrow j + 1$;
                line$[n, i] \leftarrow$ null;
            end
            else
                line$[n, i]$ = ■;
                $j \leftarrow 0$;
            end
        end
    end
end
return sequence
```

3. To each model state $ST_{[w]}$ is associated a sequence F_n:

$$F_n^{ST_{[w]}} = \{F_1, F_2, \cdots, F_k, \cdots, F_n \quad | \quad F_k \leq k, \quad \forall k \in [1, n]\} \tag{4.16}$$

which allows configuring $ST_{[w]}$ with the minimum number of translations $line[n,i]$.

4. Each sequence F_n can be linked to a corresponding $F_n^{ST_{[w]}}$, which is capable of generating the same sequence S_m:

$$C_{\mathcal{P}}[M_{\mathbb{N}} \leftarrow A_{shift}(F_n)] = C_{\mathcal{P}}[M_{\mathbb{N}} \leftarrow A_{shift}(F_n^{ST_{[w]}})] \tag{4.17}$$

where:

$$F_k^{ST_{[w]}} = F_k \bmod k, \quad \forall k \in [1, n] \tag{4.18}$$

5. Different model states $ST_{[w_1]}, ST_{[w_2]}, \ldots, ST_{[w_z]}$ are capable of generating the identical sequence S_m:

$$S_m^{ST_{[w_1]}} = S_m^{ST_{[w_2]}} = \cdots = S_m^{ST_{[w_z]}} \tag{4.19}$$

The key points discussed in considerations 3, 4, and 5 are summarised in Figure 4.1.

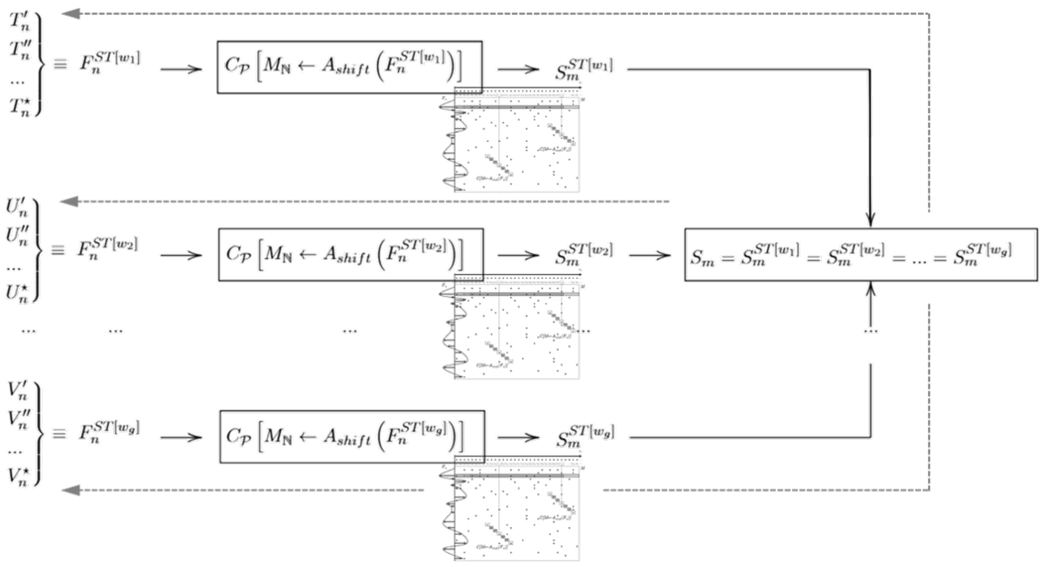

T'_n
T''_n
...
$T^\star_n$
$\equiv F_n^{ST[w_1]}$
$C_{\mathcal{P}}\left[M_{\mathbb{N}} \leftarrow A_{shift}\left(F_n^{ST[w_1]}\right)\right]$
$S_m^{ST[w_1]}$
U'_n
U''_n
...
$U^\star_n$
$\equiv F_n^{ST[w_2]}$
$C_{\mathcal{P}}\left[M_{\mathbb{N}} \leftarrow A_{shift}\left(F_n^{ST[w_2]}\right)\right]$
$S_m^{ST[w_2]}$
$S_m = S_m^{ST[w_1]} = S_m^{ST[w_2]} = \ldots = S_m^{ST[w_g]}$
V'_n
V''_n
...
$V^\star_n$
$\equiv F_n^{ST[w_g]}$
$C_{\mathcal{P}}\left[M_{\mathbb{N}} \leftarrow A_{shift}\left(F_n^{ST[w_g]}\right)\right]$
$S_m^{ST[w_g]}$

Figure 4.1 Characteristic of model $\mathcal{M}_{IS}^{\mathbb{N}}$:
$\longrightarrow$ Compute S_m up to n using $\mathcal{O}(n)$ operations from $F_n^{ST_{[w]}}$ (linear time)
$\dashleftarrow$ Compute $F_n^{ST_{[w]}}$ up to n using $\mathcal{O}(n!)$ operations from S_m (factorial time)

Conjectures

Experimentally, the sequences S_m provided in the Appendix (Table A1), as well as additional unlisted ones, have been confirmed for n values extending to approximately $n \approx 10^9$.
Based on these data, it is possible to formulate the following conjectures:

1. Specified the value n of the model $\mathcal{M}_{IS}^{\mathbb{N}}$, computing all S_m for each combination of the values of the elements of $F_n^{ST[w]}$, we observe that:

$$\exists\ a_m \in \mathbb{N} \ \mid\ a_m \neq C_{\mathcal{P}}[M_{\mathbb{N}} \leftarrow A_{shift}(F_n^{ST[w]})], \quad \forall w \in W \tag{4.20}$$

   To put it differently, there are sequences a_m that cannot be obtained from the model $\mathcal{M}_{IS}^{\mathbb{N}}$.

2. Every sub-sequence of Prime numbers $\mathcal{P}$ can be deduced from the model $\mathcal{M}_{IS}^{\mathbb{N}}$:

$$\forall S_m \subset \mathcal{P} \ \ \exists\ w_p \in W \ \mid\ S_m = C_{\mathcal{P}}[M_{\mathbb{N}} \leftarrow A_{shift}(F_n^{ST[w_p]})] \tag{4.21}$$

   For example, the sequence of Twin primes has been obtained as in Appendix (Table A2).

Both conjectures are represented in Figure 4.2. In particular, it highlights how $\mathcal{M}_{IS}^{\mathbb{N}}$ can be considered as a new generalized family of integer sequences that includes all the sequences of prime numbers. Additionally, by setting a value n and computing all the sequences S_m obtainable from the model $\mathcal{M}_{IS}^{\mathbb{N}}$, for all combinations of the possible values of $F_k^{ST[w]}$, it emerges that approximately only 30% of the sequences S_m are contained in OEIS.

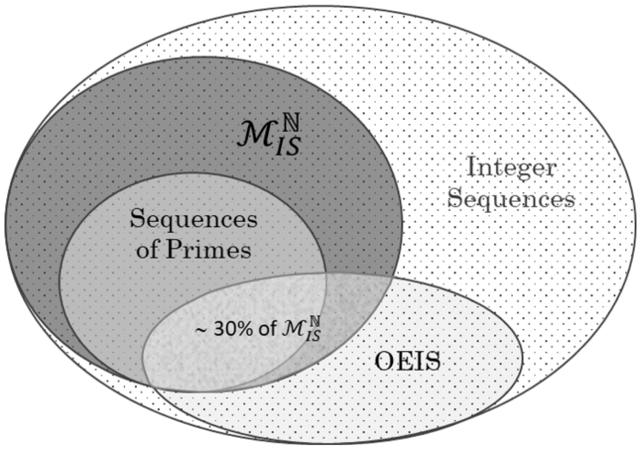

$\mathcal{M}_{IS}^{\mathbb{N}}$
Integer Sequences
Sequences of Primes
~ 30% of $\mathcal{M}_{IS}^{\mathbb{N}}$
OEIS

Figure 4.2 Conjectures on $\mathcal{M}_{IS}^{\mathbb{N}}$

Set-builder notation

Another perspective on $\mathcal{M}_{IS}^{\mathbb{N}}$ can be articulated using Set-builder notation. Beginning with the fundamental definition of Prime numbers:

$$\mathcal{P} = \{m \in \mathbb{N} \ \mid\ m \not\equiv 0 \bmod n, \quad \forall n \in [2, m-1]\} \tag{4.22}$$

we can define the sequences S_m generated by the model $\mathcal{M}_{IS}^{\mathbb{N}}$ as the set:

$$S_m = \{m \in \mathbb{N} \ \mid\ m - F_n \not\equiv 0 \bmod n, \quad \forall n \in [2, m-1]\} \tag{4.23}$$

Thus, for the sequence $F_n = \{0, 0, 0, \cdots, 0\}$, it follows that $S_m = \mathcal{P}$.

(**Prospective research**). Regarding possible future developments, which are not currently the subject of this thesis, they could involve exploring a model $\mathcal{M}_{IS}^{\mathbb{N}}$ in which an additional sequence, called G_n, is provided as input in the following way:

$$S_m = \{m \in \mathbb{N} \mid m - F_n \not\equiv 0 \bmod (G_n), \quad \forall n \in [2, m-1]\} \tag{4.24}$$

Chapter 5

Random Bit Generation

This chapter presents key concepts related to Random Bit Generation and the associated randomness tests. It not only provides definitions for these tests but also underscores their significance. After reviewing the concepts of randomness and random sequences, we delve into the functionality and characteristics of random number generators. Thereafter, the randomness tests are elaborated upon, and the strategies used to evaluate a generator's accuracy are outlined.

Subsequently, we present our proposed generator construction, grounded on the generation model $\mathcal{M}_{IS}^{\mathbb{N}}$ outlined in the previous chapter. Ultimately, its operational effectiveness will be assessed through statistical test packages and the comparison of computational performance with some of the most well-known PRNGs.

5.1 Randomness

The notion of randomness is widely recognised in everyday language, yet providing a formal definition of what constitutes randomness is a complex task. Typically, the term 'random' is connected to phenomena that defy prediction, like the selection of a winning lottery number, or appear to possess no discernible pattern or systematic behaviour within a given framework. In ancient eras, the notion of randomness was deeply connected to the idea of fate or destiny. People believed that random events, such as dice casting, controlled the outcomes and fates of individuals. As centuries passed, the word "random" began to be used within the gaming arena, a domain where its usage continues to hold significance today. It was not until the dawn of the 17th century mathematics and the subsequent development of sophisticated computational methodologies that an exhaustive examination of randomness became feasible. This progression enabled formalisation of the definition of a random number by aligning it with the concept of a random variable, thus providing a more structured understanding of randomness. Over the last few years, the concept of randomness has been recognised as a crucial element in the field of information theory, which is an area where random binary sequences are of significant importance. In this context, there are three primary methodologies that have been developed to characterise a sequence of random bits. The initial approach, proposed by von Mises, relies on the principle that the individual elements within a random string, along with its particular substrings, must exhibit a consistent frequency of occurrence.

Kolmogorov's 1963 model provided a foundational basis that establishes a connection between the notion of randomness and the principles of complexity theory. He conceptualised the randomness of a sequence by considering the minimal length of its description, which is sufficiently detailed to enable its reconstruction. Subsequently, in 1966, Martin-Löf advanced a quantitative perspective, positing that a truly random sequence should possess only a limited quantity of regular patterns and, as a result, should successfully pass specific statistical tests [She91]. It is crucial to highlight that these methodologies are not isolated, but instead exhibit significant interdependence.

5.2 Random sequences

An arbitrary sequence of bits can be regarded as analogous to the outcome of repeatedly flipping a fair coin, with one face marked 0 and the other marked 1. Because the coin is fair, each flip results in 0 or 1 with an identical probability of 1/2. The challenging endeavour lies in ascertaining whether a given sequence of bits is indeed random, as opposed to possessing some inherent pattern or predictability. To illustrate this complexity, consider, for instance, the following strings, each comprised of 41 *bits*:

$$\begin{array}{c}00000000000000000000000000000000000000000\\01101010000010011110011011000111000101001\\11011110011101011111011100110011001110100\end{array}$$

Upon initial inspection, the second and third strings might seem arbitrary, whereas the first string is often deemed questionable. Nevertheless, within the framework of probability theory, each of the three strings possesses an identical likelihood of occurrence, specifically quantified as $1/2^{41}$. Indeed, considering the entire set of possible sequences of 41 *bits*, every individual sequence is assigned the same probability of occurrence. The apparent non-random nature of the initial sequence is connected to how humans perceive randomness. Upon observing the first sequence, the brain instinctively memorizes its structure due to its straightforwardness; it consists solely of 0s, making it exceptionally easy to recollect. In contrast, the subsequent sequence appears to us to be random due to its lack of an evident pattern, posing a greater challenge to compactly retain in memory. Despite its seemingly random nature, the sequence is, in fact, the binary expansion of $\sqrt{2}-1$ and results from a deterministic mathematical process. Thus, contrary to appearance, it is not random. Interestingly, it bears a striking resemblance to a third sequence, generated purely by the stochastic method of flipping a coin. This case demonstrates two common misconceptions. Firstly, there is a tendency to assume randomness in the generation of an object merely based on its appearance. Secondly, there is a propensity to attribute chance patterns to non-random causes. Given the growing prevalence of random bit sequences in practical applications, a significant issue emerges regarding the methods necessary to produce sequences that are appropriate for a variety of specific requirements.

5.3 Random Number Generators

In this section, we elaborate on random bit generators which, as indicated by their nomenclature, are devices designed to produce sequences of random bits. Currently, there are a variety of random number generators in use, each distinguishable by an array of performance metrics and unique characteristics. The generation of random bit sequences can be achieved through multiple methodologies, broadly classified into two fundamental categories: Non-Deterministic Random Bit Generators (NDRBGs), also known as True Random Number Generators (TRNGs), and Deterministic Random Bit Generators (DRBGs), commonly known as Pseudo-Random Number Generators (PRNGs). The primary difference between these two categories is rooted in the mechanism of their generation processes.

Unpredictable

In cryptographic applications, TRNGs and PRNGs are required to produce sequences that are unpredictable. Specifically, with regard to PRNGs, even when the seed remains undisclosed, each subsequent number generated should appear random and undetermined, irrespective of the knowledge of any previously generated numbers in the sequence. This characteristic is often referred to as forward unpredictability. Additionally, it should be computationally infeasible to deduce the original seed from any number of generated values, a principle known as backward unpredictability. Furthermore, there should be no discernible relationship between the seed and any numbers produced using that seed; rather, each output in the sequence must appear to be the result of an independent stochastic process with an occurrence probability of 1/2.

To maintain forward unpredictability, it is crucial to exercise caution in acquiring seeds. Should the seed and the generation algorithm be known, the values output by a Pseudorandom Number Generator (PRNG) become entirely predictable. Given that, in numerous instances, the generation algorithm is accessible to the public, it is imperative that the seed remains confidential and cannot be deduced from the pseudorandom sequence it generates. Furthermore, the seed itself must inherently possess an element of unpredictability.

5.3.1 True Random Number Generators

A True Random Number Generator (TRNG) employs a nondeterministic source, known as the entropy source, in conjunction with a processing function known as the entropy distillation process, to generate randomness [SK14]. The implementation of a distillation process is crucial to address any potential deficiencies inherent in the entropy source that may lead to the generation of sequences lacking randomness, such as the presence of extended sequences composed solely of zeros or ones. The entropy source generally involves a physical phenomenon, like the inherent noise in an electronic circuit, the timing variations of user activities such as keystrokes or mouse movements, or the quantum mechanical effects manifested in a semiconductor material. It is common practice to use diverse combinations of these inputs to enhance the robustness and quality of the randomness produced.

Outputs generated by a TRNG can serve two purposes: they may be utilised directly as random numbers, or they can be input into a PRNG to produce further sequences. For direct usage - which implies no subsequent processing stages - the output of any TRNG must adhere to stringent criteria of randomness. These criteria are assessed through the application of statistical tests, which are designed to evaluate and confirm the apparent randomness of the physical sources that supply inputs to the TRNG. To illustrate, consider a physical source like electronic noise. Although it may seem random at a glance due to its seemingly disorderly superposition of regular patterns, such as waves or other periodic phenomena, comprehensive statistical tests may reveal an underlying non-random structure. Thus, the assessment of randomness is crucial to ensure the integrity of outputs from TRNGs.

In the context of cryptography, it is crucial that the output generated by TRNGs remain unpredictable. However, certain physical sources, such as vectors derived from date/time data, exhibit a level of predictability. To address these issues, a strategy involves blending outputs from a variety of source types to create the inputs for a TRNG. Nevertheless, even with this approach, the outputs generated by the TRNG might still fall short when subjected to rigorous statistical testing. Furthermore, the process of generating high-quality random numbers could be prohibitively time-consuming, rendering it impractical, especially when there is a demand for a vast number of random numbers. In situations requiring large volumes of random numbers, the use of PRNG could present a more viable alternative.

5.3.2 Pseudo-Random Number Generators

A PRNG relies on one or more initial inputs to produce a series of numbers that simulate randomness. These initial inputs are referred to as seeds. In scenarios where a high level of unpredictability is essential, it is imperative that the seed itself is both random and beyond prediction. Consequently, it is standard practice for a PRNG to source its seeds from an actual TRNG, which means that a PRNG requires the accompaniment of a TRNG to function correctly [STM10].

The results produced by a PRNG are generally deterministic, being defined by the initial seed; in other words, any true randomness is restricted solely to the generation of this seed. This deterministic characteristic of the process is the basis for describing it as "pseudo-random". Consequently, since every component of a pseudorandom sequence can be precisely recreated when required by using the seed from which it was derived, it is sufficient to retain only this seed if there is a need to replicate or verify the pseudorandom sequence.

Interestingly, pseudorandom numbers frequently seem more random than numbers derived from physical phenomena. When a pseudorandom sequence is meticulously designed, each number in the sequence is generated based on its predecessor through transformations that seem to enhance randomness. A succession of these transformations can effectively remove statistical auto-correlations between input values and the resulting output. Consequently, the outputs of a PRNG can possess superior statistical attributes and be generated more rapidly than those of a TRNG.

5.3.3 Properties

The essential attribute of a PRNG lies in its ability to output sequences of numbers that give the impression of being random. In practical terms, it is crucial to ensure that the sequences generated adhere to the criteria typically expected from truly random sequences. It is practically unfeasible to provide a theoretical proof that a given generator produces genuinely random sequences. Consequently, to assess the uniformity and randomness of the output of a PRNG, comprehensive statistical assessments are conducted on the sequences it generates.

To facilitate this evaluation process, a myriad of statistical tests are available that proficiently examine the randomness of these sequences. Each statistical test is designed to scrutinise a particular attribute of the sequence, determining whether it conforms to the expectations of a uniform distribution. For example, when analysing a binary sequence composed of n *bits*, where n is sufficiently large, it is expected that the frequency of $0s$ should closely approximate $n/2$, similarly, the same is expected for the frequency of $1s$. This criterion is a fundamental expectation when one assumes that the sequence is uniform in nature.

When statistical evidence is presented against the expected outcome, the sequence in question will fail the test. It is crucial to note that this attribute cannot be effectively assessed using only a single sequence. For the assessment of randomness to be deemed robust and reliable, it is imperative to administer the test on a statistically significant sample comprising multiple sequences. The generator quality is then evaluated on the basis of the percentage of sequences that successfully pass the test. It is important to recognise that a generator, even if it generally produces high-quality output, may sporadically produce sequences that deviate from the anticipated proportion of 0 and 1.

To accurately evaluate the performance of a generator, it is crucial to perform the test on a sufficiently large sample of sequences generated by it. Detailed information on randomness tests can be found in Section 5.4. Beyond this essential criterion, several other attributes also play a significant role in defining the generator's quality. Some of the key parameters include the following:

– *Period*: Considering that the internal state space of a PRNG initialised with a seed is finite, the generator will eventually revisit its initial state, forming a cycle. Since output bits are produced deterministically from the internal state, the outputs will inevitably repeat. The shortest cycle length before this repetition is termed the PRNG period. Clearly, a longer period is indicative of a higher-quality generator. Ideally, the period should closely match the total number of unique internal states available to the generator.

– *Efficiency*: The evaluation of a PRNG's effectiveness involves analysing its memory consumption and processing speed. An efficient generator is expected to utilise a minimal amount of computational resources.

– *Repeatability*: If initialised identically, pseudorandom number generators (PRNGs) should consistently produce an identical sequence of random bits.

– *Portability*: Ensuring a PRNG is architected to operate independently of specific hardware and software is essential, allowing its deployment and utilisation in diverse environments.

Cryptographically Secure Pseudorandom Number Generator (CSPRNG)

Besides the previously mentioned features, a generator used in cryptographic contexts must satisfy additional requirements to be classified as a Cryptographically Secure Pseudorandom Number Generator. This fact leads to the question of how to design PRNGs that are unpredictable in reasonable (i.e., polynomial) time. In response to this question, the concept of cryptographically secure PRNGs was first proposed by [BM84; Yao82]. A pseudorandom number generator (PRNG) is deemed cryptographically secure if it withstands all statistical assessments executable in polynomial time, meaning no polynomial time algorithm can distinguish the bit sequences it produces from those generated by a true random process. Conversely, a PRNG is termed polynomially predictable if there is a polynomial time algorithm capable of predicting its subsequent outputs, given a sufficiently extensive sequence of its generated *bits*.

Definition 5.3.1. *(Pseudorandom Generator): A pseudorandom generator is a deterministic polynomial-time algorithm G satisfying the following two conditions:*

1. *(Expansion) There exists a function $l : \mathbb{N} \to \mathbb{N}$ such that $l(n) > n$ for all $n \in \mathbb{N}$, and $|G(s)| = l(|s|)$ for all $s \in \{0,1\}^*$.*
2. *(Pseudorandomness) The ensemble $\{G(U_n)\}_{n \in N}$ is pseudorandom.*

The function 1 is called the expansion factor of G [Gol01; KL21].

The input s to the generator is its its seed. The expansion condition requires that the algorithm G map n-bit-long seeds into $l(n)$-bit-long strings, with $l(n) > n$. The pseudorandomness condition requires that the output distribution induced by applying algorithm G to a uniformly chosen seed be polynomial-time-indistinguishable from a uniform distribution, although it is not statistically close to uniform. Specifically, we can bound the statistical difference between $G(U_n)$ and $U_{l(n)}$ as follows:

$$\begin{aligned}\frac{1}{2} \cdot \sum_x \left|\Pr\left[U_{l(n)} = x\right] - \Pr\left[G(U_n) = x\right]\right| &= \max_S \left\{\Pr\left[U_{l(n)} \in S\right] - \Pr\left[G(U_n) \in S\right]\right\} \\ &\geq \Pr\left[U_{l(n)} \notin \{G(s) : s \in \{0,1\}^n\}\right] \\ &\geq \left(2^{l(n)} - 2^n\right) \cdot 2^{-l(n)} \\ &= 1 - 2^{-(l(n)-n)} \geq \frac{1}{2}\end{aligned} \tag{5.1}$$

where the last inequality uses $l(n) \geq n+1$. Note that for $l(n) \geq 2n$, the statistical difference is at least $1 - 2^{-n}$.

The foregoing definition is quite permissive regarding the expansion factor $l : N \to N$. It asserts only that $l(n) \geq n+1$ and $l(n) \leq \text{poly}(n)$. (It also follows that $l(n)$ is computed in time polynomial in n; e.g., by computing $|G(1^n)|$.) Clearly, a pseudorandom generator with expansion factor $l(n) = n+1$ is of little value in practice, since it offers no significant saving in coin tosses. Fortunately, as shown in the next subsection, even pseudorandom generators with such a small expansion factor can be used to construct pseudorandom generators with any polynomial expansion factor. Hence, for every two expansion factors $l_1 : N \to N$ and $l_2 : N \to N$ that can be computed in poly (n) time, there exists a pseudorandom generator with expansion factor l_1 if and only if there exists a pseudorandom generator with expansion factor l_2. This statement is proved by using any pseudorandom generator with expansion factor $l_1(n) \overset{\text{def}}{=} n + 1$ to construct, for every polynomial $p(\cdot)$, a pseudorandom generator with expansion factor $p(n)$. Note that a pseudorandom generator with expansion factor $l_1(n)$ can be derived from any pseudorandom generator.

Each pseudorandom generator will have a predetermined expansion function. We shall consider "variable-output pseudorandom generators" that, given a random seed, will produce an infinite sequence of bits such that every polynomially long prefix of it will be pseudorandom.

To resist cryptanalysis, a CSPRNG should remain secure even if an adversary gains access to portions of its initial or subsequent states [Yao82; BM84]. The essential characteristics are as follows:

– *Prediction Resistance*: Ensure that a breach in the DRBG's internal state does not compromise the security of future outputs is paramount. If an adversary becomes aware of the internal state at iteration x and understands the PRNG's operation, they can predict future states from $x + 1$ onwards. To prevent this vulnerability, it is necessary to refresh the seed by incorporating a new entropy with each generation request, which matches the required security level. When new entropy is limited, the attacker cannot ascertain subsequent states easily, yet they face reduced but finite possibilities. Thus, the post-compromise security hinges on the quantity of fresh entropy. A DRBG should consistently access entropy sources and integrate new entropy into its process whenever feasible.

– *Backtracking Resistance*: The assurance that the output sequences of a generator remain indistinguishable from perfect random sequences, even if the PRNG becomes compromised at a later stage. In practical terms, backtracking resistance guarantees that an adversary, even if in possession of state x, cannot tell the difference between the outputs generated from states $1, 2, \ldots, x-1$ and truly random outputs, nor can they reconstruct earlier states. This property is achieved by designing the generation algorithm as a one-way function. A one-way function, or unidirectional function, is computationally easy to perform but challenging to reverse. This signifies that from the internal state x, transitioning to the next state $x + 1$ is straightforward. Nonetheless, given the state $x + 1$, there is no polynomial-time algorithm that can determine the preceding state x.

A CSPRNG is consequently constructed to resist cryptanalysis. It is crucial to note that while every CSPRNG is a type of PRNG, the inverse does not hold. Some PRNGs may succeed in all randomness evaluations yet remain susceptible to reverse engineering attacks. For instance, consider the Linear Congruential Generator

(LCG) described in Section 3.1.3, a widely recognized generator often employed in simulations, which, due to its predictability, is entirely unsuitable for cryptographic applications [Kra86; Boy89a; Boy89b].

5.3.4 Standards

Nel 2010, il National Institute of Standards and Technology (NIST) ha prodotto una serie di raccomandazioni su come si dovrebbe progettare un buon generatore di numeri casuali.

The project provides guidelines through the Special Publication (SP) 800-90 series, which includes recommendations on deterministic random bit generator (PRNG) mechanisms, entropy sources, and construction principles for TRNGs, and has three parts:

- SP 800-90A, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* [RBG15], specifies several approved PRNG mechanisms based on approved cryptographic algorithms that, once provided with seed material that contains sufficient randomness, can be used to generate random bits suitable for cryptographic applications. NIST is revising SP 800 90A to be consistent with SP 800-90C.
- SP 800-90B, *Recommendation for the Entropy Sources Used for Random Bit Generation* [RBG18], provides guidance for the development and validation of entropy sources, which are mechanisms that generate entropy from physical or non-physical noise sources and that can be used to generate the input for the seed material needed by a PRNG or for input to an TRNG.
- SP 800-90C, *Recommendation for Random Bit Generator (RBG) Constructions* [RBG24], specifies constructions for the implementation of TRNGs. SP 800-90C also provides high-level guidance for testing TRNGs for conformance to this recommendation.

The Figure 5.1 explains the relationship of the three parts of the series:

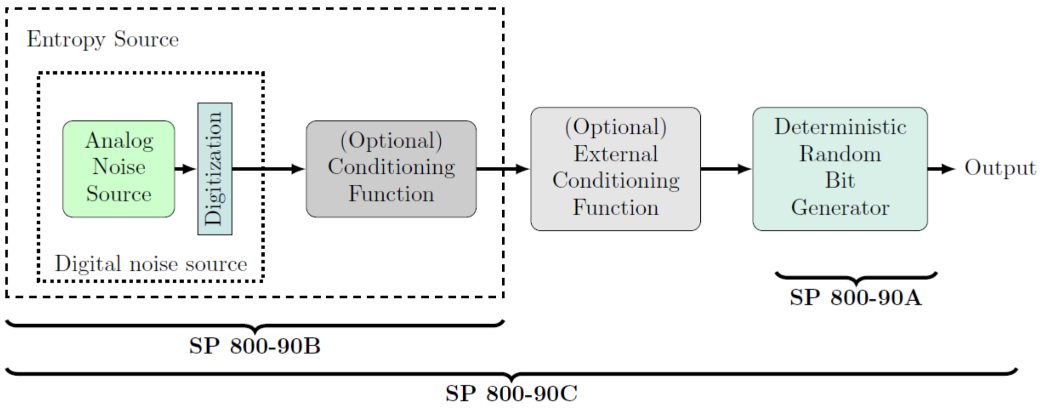

Entropy Source
Analog Noise Source
Digitization
(Optional) Conditioning Function
Digital noise source
(Optional) External Conditioning Function
Deterministic Random Bit Generator
Output
SP 800-90A
SP 800-90B
SP 800-90C

Figure 5.1 NIST Special Publication 800-90 series.

Furthermore, NIST Interagency Report (IR) 8427 *Discussion on the Full Entropy Assumption of the SP 800 90 Series* provides technical discussions to support the full entropy definition used in the SP 800 90 series [RBG23].
The influence exerted by these publications has been significant, as prior to their release, there was an absence of established guidelines for the correct construction of a generator. As a result, many of the commonly used generators had security measures that were insufficient for the purposes they were intended to serve.

5.4 Randomness test

To confirm the efficacy of a generator, it is necessary to check its indistinguishability from an ideal generator. A fundamental criterion for a PRNG is its ability to generate sequences that cannot be differentiated from truly random ones. To evaluate if the generator in question meets this criterion, statistical evaluations, commonly referred to as randomness tests, are conducted to detect specific types of vulnerabilities the generator might exhibit. In an ideal scenario, to evaluate the distribution of generated sequences, we would utilise a single test to confirm that every sequence generated by the algorithm is equally likely. Given that the total number of potential sequences is 2^n, where n denotes the length of the sequence in bits, this test becomes impractical as n grows larger. To determine the randomness of a PRNG, a rigorous randomness analysis is conducted on its outputs. This involves applying various statistical tests to the generated sequences. These tests assess whether the sequences exhibit specific characteristics expected from genuinely random sequences. Each test searches for "patterns," and their detection suggests non-randomness in the sequence. The literature offers hundreds of established statistical tests that can be employed to evaluate a generator's randomness. A solitary test is insufficient to ascertain the randomness of a sequence, owing to the possibility of various forms of non-randomness. Nevertheless, if a generator successfully completes numerous diverse tests, the confidence in its randomness is enhanced. Consequently, compilations of multiple tests, commonly termed test suites or randomness test batteries, have proliferated in the literature. Each test in the suite is designed to assess a different feature of randomness. Should all tests or a particular fraction of them affirm that the PRNG generates random numbers, the PRNG is regarded as high-quality. The challenging issue lies in deciding which tests to perform, how many to conduct, and determining the minimum threshold of passing tests relative to the total implemented tests, so as to regard the generator as high-quality. Considering the possibility of infinitely many tests, all equally legitimate, no finite collection of tests can be deemed "complete." Any finite selection of tests inevitably fails to capture certain defects that might exist in a generator. Hence, as previously noted, using all literature-known tests would still not assure absolute certainty of the generator's randomness. In practical settings, resources are finite; thus, it becomes essential to choose a practical number of tests. Typically, the selection is guided by the specific requirements of the application employing the generator. Another challenging issue is deciding which tests to incorporate into a suite, presupposing the existence of superior tests. Nevertheless, as previously mentioned, all tests should be considered equally valid. Each test provides a means to assess a generator's randomness, and hence, irrespective of the feature evaluated by the test, a generator producing genuinely random numbers ought to pass it.

5.4.1 Hypothesis test

The statistical evaluation of a random generator generally involves the use of one or more hypothesis tests on the sequences it produces. These tests are a widespread type of statistical analysis employed in numerous applications to assess a specific hypothesis from the observations of experimental data. The results of these tests indicate how well the observed data align with the predetermined hypothesis.

Within the framework of a hypothesis test, a random experiment is contemplated in which samples are drawn at random from a specified sample space. In alignment with the test's objective, two hypotheses are articulated concerning the underlying distribution of the sample drawing process, referred to as the data distribution:

– The null hypothesis, denoted as H_0, represents the proposition subject to verification. It imposes a condition on a parameter of the data distribution (such as mean or variance) or directly on the distribution type (e.g. normal, Poisson, uniform, etc.). In other words, the null hypothesis encompasses a collection of potential data distributions that satisfy the proposed statement. The null hypothesis is classified as simple if it specifies a single data distribution; otherwise, it is termed composite.

– The alternative hypothesis (H_A) refers to the proposition that stands in contrast to the null hypothesis. Although it may theoretically correspond to a distinct data distribution, it is generally characterised as the opposite of the null hypothesis, comprising the infinite set of data distributions that are complementary to the null hypothesis.

When a data sample is randomly selected from the sample space, the test aims to either confirm H_0 or reject it in favor of H_A by comparing the observed sample with the anticipated data from the theoretical model associated with H_0. If the data aligns well, H_0 is supported; otherwise, it is dismissed in favor of H_A. In this context, alignment implies that any discrepancies between the observed and expected data can be ascribed to random variation (in line with a predefined criterion) and do not indicate a fundamental distinction between the two datasets.

Methodologically, the null hypothesis is presumed true unless the experimental data, assessed via hypothesis testing, suggest that the alternative hypothesis holds. Therefore, accepting the null hypothesis essentially signifies that the tests fail to provide adequate evidence to substantiate the alternative hypothesis.

Methodology

Typically, when presented with a data sample, performing a hypothesis test involves the following procedure. A numerical value, termed the test statistic, is derived from the given data sample. A decision is then reached by contrasting this test statistic against a pre-established acceptance region (commonly known as a confidence interval, consisting of test statistic values for which H_0 is accepted) and a rejection region (comprising values where H_0 is rejected, thereby accepting H_A). The methodology by which a particular test statistic is linked to the data and the manner in which the acceptance and rejection regions are predetermined define the specific hypothesis test.

From a formal perspective, it is important to note that the data sample, prior to observation, must be regarded as random. Consequently, as the test statistic processes a random sample and outputs a numeric result, it can be represented as a random variable.

Output

Considering the null hypothesis H_0 and the alternative hypothesis H_A, evaluation with a data sample yields two potential conclusions from the test application:

– either to accept H_0 (and refuse H_A).

– or to refuse H_0 (and accept H_A).

Table 5.1 illustrates four potential configurations based on the test's output and the genuine state of H_0 (either true or false), a variable that remains unknown. The primary objective of a hypothesis test is, in fact, to deduce conclusions about this unknown state.

TRUE SITUATION	CONCLUSION	
	Accept H_0	Accept H_a (reject H_0)
Data is random (H_0 is true)	No error	Type I error
Data is not random (H_a is true)	Type II error	No error

Table 5.1 Test configurations used to determine the acceptance or rejection of the null hypothesis H_0.

The different configurations, arising from the validation or erroneous conclusion of the test, are depicted below:

– The null hypothesis is true (Data is random):

  $\mapsto$ and the test validates it (No error).

  $\mapsto$ but the test refutes it (false positive): this type of error is referred to as a Type I Error. The likelihood of a Type I Error occurring is typically represented by α. When the null hypothesis is dismissed, we state that the test result is statistically significant at a significance level of α.

– The null hypothesis is false (Data is not random):

  $\mapsto$ and the test refutes it (No error).

  $\mapsto$ but the test accepts it (false negative): this kind of error is referred to as a Type II error. The likelihood of incurring a Type II error is commonly represented by β. Moreover, $1-\beta$ is termed the test power, which denotes the probability that the test accurately favours the alternative hypothesis.

In relation to this, we can discuss with respect to the parameters α and β:

- The value α tied to a test indicates the probability of obtaining false positives. This occurs when the test statistic falls within the rejection region, even though the null hypothesis is correct, meaning that the sample is drawn from the sample space in accordance with the data distribution.
- In contrast, the parameter β denotes the likelihood of obtaining false negatives, which occurs when the test statistic falls within the acceptance region even though the null hypothesis is incorrect. If the alternative hypothesis is accurately specified by a distinct probability distribution, then β can be calculated. It can be shown that an increase in α results in a decrease in β, and the opposite is also true.

The entire process is illustrated in Figure 5.2.

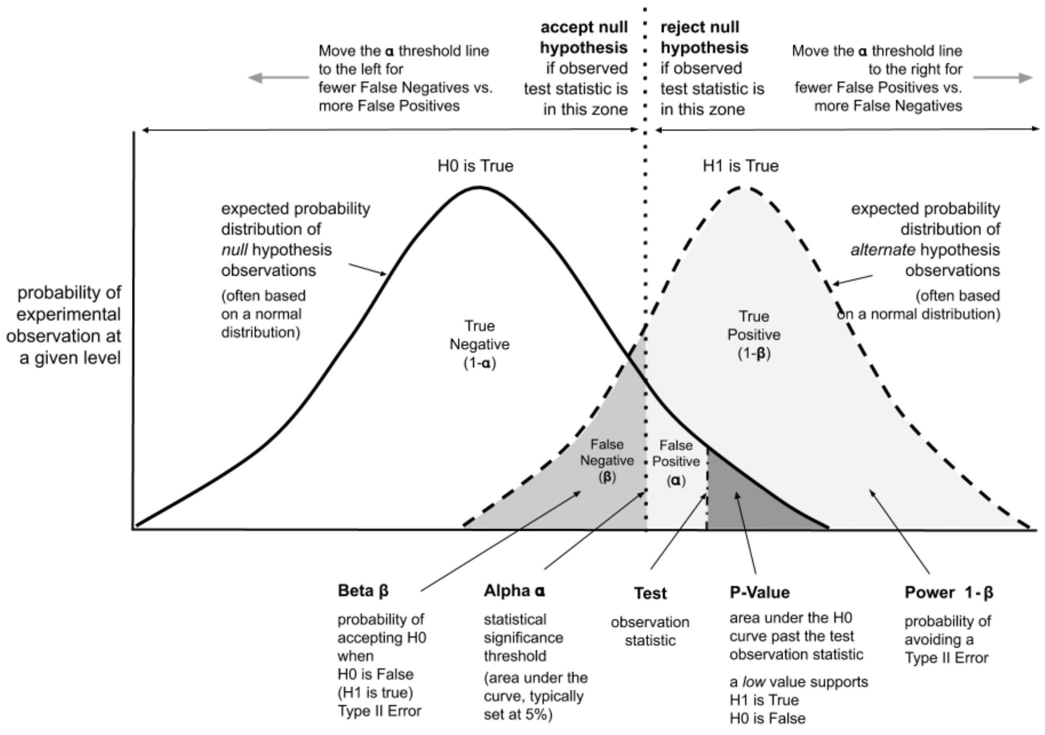

Move the α threshold line to the left for fewer False Negatives vs. more False Positives
accept null hypothesis if observed test statistic is in this zone
reject null hypothesis if observed test statistic is in this zone
Move the α threshold line to the right for fewer False Positives vs. more False Negatives
H0 is True
H1 is True
expected probability distribution of null hypothesis observations
(often based on a normal distribution)
expected probability distribution of alternate hypothesis observations
(often based on a normal distribution)
probability of experimental observation at a given level
True Negative (1-α)
True Positive (1-β)
False Negative (β)
False Positive (α)
Beta β
probability of accepting H0 when H0 is False (H1 is true) Type II Error
Alpha α
statistical significance threshold
(area under the curve, typically set at 5%)
Test
observation statistic
P-Value
area under the H0 curve past the test observation statistic
a low value supports H1 is True H0 is False
Power 1-β
probability of avoiding a Type II Error

Figure 5.2 Hypothesis testing. Set of all possible observations (H_0, H_1).

.

5.4.2 Evaluation methods

Given the test statistic of an observed data sample, in order to decide if the null hypothesis H_0 has to be accepted or rejected (or, equivalently, to define the acceptance region and the rejection region) two methods are typically used:

- The critical value method involves establishing one or more critical value(s) that serve as thresholds to demarcate the acceptance region from the rejection region. Subsequently, for each sample of data, the null hypothesis H_0 is rejected if the test statistic is equal to or exceeds the critical value in the direction favoring the alternative hypothesis (as further detailed later); if not, the null hypothesis is accepted.

– The p-value method relies on the critical value method, adding a probabilistic interpretation in order to make the testing process more friendly and the results easier to interpret.

Specifically, the p-value represents the probability, under the assumption that the null hypothesis holds true, of observing a test statistic as extreme as, or more extreme than, the value obtained from the experiment. A decision is made by choosing a threshold value α close to 0. If the p-value is less than or equal to α, the null hypothesis is rejected; conversely, if the p-value surpasses α, the null hypothesis is not rejected. A smaller p-value suggests stronger evidence against the null hypothesis. It can be shown that:

– The significance level α corresponds to the probability of making a Type I error: $\mathbf{P}(\text{reject } H_0 \mid H_0 \text{ is true}) = \alpha$.

– The probability of a Type II error is denoted by $\beta = \mathbf{P}(\text{accept } H_0 \mid H_0 \text{ is false})$.

Typically, the likelihood of committing Type I errors is set at a constant value. In cryptography, standard values for α fall within the interval $[0.001; 0.01]$. In contrast to α, the parameter β is not constant and can assume a multitude of values, since there are infinite scenarios where the hypothesis H_0 may be false.
The two evaluation methods operate with two different approaches. Basically, they operate within distinct domains. The critical value approach functions within the range of potential test statistic outcomes, with the acceptance and rejection regions delineated by the critical value(s). Conversely, the p-value approach works within the probabilistic domain, specifically the interval $[0, 1]$, where the acceptance and rejection regions are determined by the parameter α.

5.4.3 Testing strategy and Result interpretation

To assess the randomness of a generator, randomness tests, which serve as hypothesis tests, are typically employed. In these tests, the null hypothesis posits that the sequence in question is random, whereas the alternative hypothesis contends that the sequence is not random. Should empirical evidence suggest that H_0 is incorrect, the null hypothesis is rejected in favor of the alternative. Testing the randomness of a PRNG involves subjecting a collection of its sequences to statistical tests and analyzing the proportion that succeeds or fails. A sequence is deemed to pass the randomness test if its p-value meets or exceeds the predetermined significance level. Assuming the generator yields random bits, it is anticipated that around $1 - \alpha$ of the sequences will pass, with the rest α failing. Hence, even when a generator functions correctly, the inherent nature of statistical testing implies some sequences will not pass. The number of failing sequences aligns with α, as this represents the probability of incurring Type I errors.

Result interpretation

When subjecting a series of n sequences, each with a constant length m, to a randomness test, a corresponding set of n p-values is obtained. By using a specified significance level, α, for the randomness test, we can assess the test results for each sequence. A sequence is considered to pass the test if its p-value is $\geq \alpha$; otherwise, it fails. Consequently, a vector $S = (s_1, \ldots, s_n)$ of length n, where each element s_i is either True or False. The True value indicates the i-th sequence passed the randomness test, while False denotes a failure in passing. The NIST documentation provides two distinct methods for assessing and interpreting the results obtained.

– *Examine the proportion of sequences that pass a statistical test.* We perform a statistical analysis to determine if the empirically calculated proportion of sequences passing a randomness test, denoted $\hat{p}$, is statistically consistent with the expected proportion $1 - \alpha$. If the proportion of success-sequences falls outside of following acceptable interval, there is evidence that the data is non-random:

$$\hat{p} \pm 3\sqrt{\frac{\hat{p}\,(1-\hat{p})}{n}} \tag{5.2}$$

where $\hat{p} = 1 - \alpha$ and n is the number of sequences. This interval is determined to be 99.73% range of normal distribution which is an approximation of the binomial distribution under the assumption that each sequence is an independent sample. In cryptographic applications, where α is typically $\alpha = 0.01$, applying this approximation requires a minimum of 1000 sequences.

– *Check the distribution of the p-values.* According to statistical theory, when conducting a hypothesis test with a valid null hypothesis and an invertible cumulative distribution of the test statistic, the p-values are uniformly distributed across $[0, 1]$. In a randomness test, therefore, if the sequences are genuinely random and the test statistic is continuous, the resulting p-values should exhibit a uniform distribution throughout the interval $[0, 1]$. Assuming that the conditions ensuring the uniform distribution of p-values are satisfied, a suggested approach to verify their uniformity over $[0, 1]$ is to partition the interval from 0 to 1 into 10 equal parts and allocate p-values to these sub-intervals based on where they fall. Following this categorisation, a chi-square test is conducted to evaluate whether the number of p-values in each sub-interval is evenly distributed. The chi-square test "χ" is a common statistical method to test whether the observed frequencies correspond to the expected frequencies of a specific probability distribution. In this scenario, given that the p-values should follow a uniform distribution, the expected frequency for each of the 10 intervals should be $n/10$, where n denotes the total number of p-values. We define the vector F, where F_i, for $i = 1 \ldots 10$, represents the observed frequency in the i-th sub-interval, and the chi-square statistic is then expressed as:

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - n/10)^2}{n/10} \tag{5.3}$$

The distribution of this statistic is a chi-square with $\nu = 9$ degrees of freedom.

In order to confirm that the generator performs the statistical test, it is crucial that the two evaluations mentioned above align with the anticipated theoretical results. The following sections delve into the rationale and construction of these essential checks in detail.

5.4.4 Statistical test suite

It is important to note that the test statistic related to a specific test serves as a numerical indicator that summarizes the sample data. Naturally, since it represents a synthesis, it inherently captures only certain characteristics of the data in question.

The range of statistical irregularities detectable by the test is dictated by the test statistic, which generally limits its ability to identify other deviations from the null hypothesis. For instance, in evaluating a random binary sequence, a test statistic that tallies the number of 1s can effectively assess whether the sequence follows the balancing criterion (i.e., the expected number of 0s matches the expected number of 1s). However, it typically does not facilitate the detection of other anomalies, such as correlations between consecutive bits. Consequently, the utility of the test in yielding meaningful insights is contingent upon whether the detected properties are pertinent to the specific application context.

To enhance the efficacy of analysis, we often employ test suites (also referred to as collections or batteries) comprising multiple varied tests designed to capture a wide range of statistical anomalies. Nevertheless, it remains impossible for any test suite to detect every conceivable anomaly.

Thus, balancing the tests within a suite involves weighing the efficiency of their implementation against their capacity to identify significant anomalies. Fundamentally, our aims are to:

1. To decrease the costs associated with implementation, it is advisable to reduce the number of tests conducted.

2. To enhance their efficacy, defined as the capacity to identify and delineate the non-random characteristics pertinent to the particular application scenario, it is essential. Based on the given use case, the analyst may prioritise certain statistical properties over others due to their relevance and importance to the analysis objectives.

3. Opt for independent tests. It is important to note a nuanced aspect in defining a test suite: the independence of selected tests. Informally, two tests are said to be independent if the outcome of one does not offer any insight into the outcome of the other. Ensuring test independence within a battery is advantageous for two primary reasons. To begin with, independent tests each offer new insights into the validity of the null hypothesis, whereas dependent tests may lead to the inefficient use of computational resources, yielding redundant information. Furthermore, and perhaps more critically, independence facilitates the derivation of more accurate conclusions from the application of multiple tests.

In the literature, several suites have been proposed and are commonly used to test the randomness quality of a given generator. The suites differ in many aspects, such as the choice of the tests, the efficiency of implementation, their flexibility, and the user interface [LV21]. Presented below are the most well-known statistical test suites, organized according to their chronological introduction in the scientific community:

– *Knuth*: the initial comprehensive suite, introduced in 1969 [Knu69], encompasses a fundamental collection of 11 tests and is recognised as the forerunner in systematic randomness testing. Although it focusses more on real numbers, making it less ideal for integer (binary) sequences, its primary mention here is for historical context.

– *Crypt-X*: the battery was presented in 1992 by the Information Security Research Centre at Queensland University of Technology for commercial purposes [Cae92]. This battery includes 6 tests: Binary derivative, Change point, Frequency, Linear complexity, Runs and Sequence complexity. Crypt-X supports stream ciphers, block ciphers, and keystream generators.

– *Diehard*: the test battery launched in 1995 [Mar95] continues to serve as a valuable resource within the statistical community, although with constraints regarding sample size and accessibility for users. A review of this battery is discussed in [Ala10].

– *SPRNG*: released in 2020, the test suite for the Scalable Parallel Random Number Generators Library (SPRNG) includes both statistical and physically-based tests [MS00]. Statistical tests are structured so that the anticipated value for a specific test statistic is predefined for an independent, identically distributed random sample taken from a uniform distribution.

– *NIST Statistical Test Suite (STS)*: First introduced by NIST in 2001 [RBG10], this suite has become the accepted standard for evaluating binary random sequences, particularly with an emphasis on applications in cryptography. Detailed examinations and evaluations of the NIST suite have been discussed in [KUH04; PRS12; MS15].

– *Dieharder*: an expanded edition of Diehard, featuring extra tests, was presented in 2006 [BEB06]. A recent critique of the implementation of the suite has been published in [SOMK22].

– *Gjrand*: released in 2006, the developed suite not only comprises a list of PRNGs, but also includes a comprehensive list of statistical tests [Jon06].

– *TestU01*: initially introduced by [LS07], this comprehensive package is notably adaptable and features an efficient design, first proposed in 2007. For a detailed assessment of the suite, see [McC06].

– *ENT*: alternative and less commonly used batteries include the ENT battery, presented in 2008 [Wal08]. ENT is designed to evaluate systems for simulation and cryptographic applications. The ENT battery operates in two modes: binary and byte, and depending on the selected mode, it computes and presents distinct statistical analyses.

– *PractRand*: a practical and adaptable suite [DH10] presented in 2010, which is increasingly being recognised in the research field. For a comparative analysis between TestU01 and Practrand, refer to [SC20].

NIST Statistical Test Suite

The NIST suite comprises 15 empirical tests adapted to evaluate binary sequences (bit-streams) [RBG10]. These tests assess randomness based on different statistical properties of individual bits or blocks of bits as described in Table 5.2. Every test within the NIST STS suite inspects the overall randomness in the bit-stream. Additionally, some tests are capable of identifying localised nonrandomness by segmenting the bitstream into multiple, generally large segments and calculating bit characteristics for each segment. These individual characteristics are then used to compute the test statistic.

Test	Statistical property analysed
Frequency (Monobit)	Proportion of 0s and 1s
Frequency within a Block	Proportion of 1s within bit blocks
Runs	Total number of runs (uninterrupted sequence of identical bit)
Longest Run of Ones in a Block	Longest run of ones within blocks
Binary Matrix Rank	Rank of disjoint sub-matrices
Discrete Fourier Transform (Spectral)	Peak heights in the Discrete Fourier Transform
Non-overlapping Template Matching	Occurrences of given aperiodic patterns
Overlapping Template Matching	Occurrences of pre-specified target strings
Maurer's Universal Statistical	Number of bits between matching patterns
Linear Complexity	Length of a linear feedback shift register (LFSR)
Serial	Frequency of all possible overlapping patterns
Approximate Entropy	Frequency of repeating patterns in the string
Cumulative Sums (Cusums)	Maximal excursion of the random walk defined by the cumulative sum of adjusted (-1, +1) digits
Random Excursions	Number of cycles having exactly K visits in a cumulative sum random walk
Random Excursions Variant	Total number of times that a particular state is visited (i.e., occurs) in a cumulative sum random walk

Table 5.2 Statistical properties analyzed in NIST STS.

Each NIST STS assessment is outlined by the test statistic pertaining to one of the following three categories, and it evaluates the randomness of the sequence based on:

1. *bits* - these tests examine various properties of bits, including bit proportion, frequency of bit alternation (runs), and cumulative sum distribution.

2. *m-bit blocks* - these tests evaluate the distribution of m bit blocks, where m generally comprises less than 30 bits within the sequence or its subdivisions.

3. *M-bit parts* - these tests examine intricate characteristics of M-bit segments (where M commonly exceeds 1000 bits) of the sequence, such as the rank of the sequence when regarded as a matrix, the sequence's spectral properties, or the linear complexity of the bitstream.

Tests are parameterized based on n, which represents the bit-length of the binary sequence subject to testing. Some tests additionally use a second parameter, indicated as m or M. Since the reference distributions of the NIST STS test statistics are approximated using asymptotic distributions (such as χ^2 or normal), the accuracy of the resulting p-values is ensured only for specific parameter values. Table 5.3 outlines the parameter values recommended by NIST for each specific test.

Test #	Test name	n	m or M	# sub-tests
1.	Frequency (Monobit)	$n \geq 100$	-	1
2.	Frequency within a Block	$n \geq 100$	$20 \leq M \leq n/100$	1
3.	Runs	$n \geq 100$	-	1
4.	Longest run of ones in a Block	$n \geq 128$		1
5.	Binary Matrix Rank	$n > 38912$	-	1
6.	Discrete Fourier Transform (Spectral)	$n \geq 1000$	-	1
7.	Non-overlapping T. M.	$n \geq 8m - 8$	$2 \leq m \leq 21$	148*
8.	Overlapping T.M.	$n \geq 10^6$		1
9.	Maurer's Universal	$n > 387840$		1
10.	Linear complexity	$n \geq 10^6$	$500 \leq M \leq 5000$	1
11.	Serial		$2 < m < [\log_2 n] - 2$	2
12.	Approximate Entropy		$m < [\log_2 n] - 5$	1
13.	Cumulative sums	$n \geq 100$		2
14.	Random Excursions	$n \geq 10^6$		8
15.	Random Excursions Variant	$n \geq 10^6$		18

Table 5.3 NIST STS configurations. For every specific test, the suggested size of the bitstream is denoted as n. Certain tests involve a secondary parameter, indicated as m or M, depending on the context. The table provides recommended configurations for this secondary parameter and enumerates the number of sub-tests associated with each test.

Multiple tests from the NIST STS battery are conducted in various forms, meaning they include multiple sub-tests to assess additional characteristics of sequences of the same type. For example, the Cumulative Sum test evaluates a sequence using both forward and backward cumulative sums. Table 5.3 provides a summary of the number of sub-tests associated with each specific test. The Non-overlapping template matching test is marked by an asterisk since the number of its sub-tests is not fixed and depends on the value chosen for the parameter m (the number 148 mentioned in Table 5.3 corresponds to the default value of the parameter $m = 9$).

(**Testing**). NIST STS provides the capability to evaluate an input file either as a singular sequence or by segmenting it into sequences of a predetermined length n, which is defined via the command line. Users must select parameters that are presented sequentially in the text-based interface as follows:

1. *File for the analysis* - the user has the option to select their own file or generate data using one of the established pseudorandom number generators, such as Blum-Blum-Shub, various congruential generators, modular exponentiation, among others.

2. *Tests* - which statistical test or tests should be utilized for the data.

3. *Values for the second parameter (m or M) for several tests* - block frequency test uses 128 blocks, Non-overlapping and Overlapping template matching both utilise 9 templates, Approximate entropy takes 10 elements, Serial test deploys 16 elements, and Linear Complexity evaluates 500 sequences (default parameters are indicated within brackets following each test name).

4. *Number of bitstreams* - to be processed.

5. *File format* - either in ASCII format, where the sequence consists of ASCII 0's and 1's, or in binary format, in which each byte of the file is made up of 8 bits from the sequence.

The selection of appropriate tests for the analysis of randomness poses a challenging question. The choice hinges on the generator (data) in question, its specific application domain, and unacceptable randomness defects. In the absence of detailed information about the data under analysis, it is recommended to employ all NIST STS tests for evaluating randomness: $< n$ (the bitlength of the sequences) must exceed 100,000 for the complete test application, as detailed in Table 5.3. According to the NIST STS guidance, a minimum of $k = \alpha^{-1} = 100$ sequences is recommended for testing. This value is suitable for the uniformity test of p-values, with a minimum of 55 sequences that need to be evaluated. The NIST STS utilises approximations in p-value processing; therefore, testing a larger number of sequences enhances the accuracy of the results. NIST recommends conducting tests on 1000 or more sequences.

5.5 Our construction

We constructed a PRNG by adopting a model outlined in [RBG15], specifically utilizing the CTR-DRBG configuration. As is known, this mode uses the encryption of an incrementing counter under a block cipher to generate outputs. The block cipher may be either 3DES with a 64-bit key or AES with a key of length 128, 192, or 256 bits. The design mixes in additional data at various stages. A derivation function (commonly the same block cipher under a different key) can optionally be used to extract entropy from the additional data. In this instance, we propose adapting the CTR-DRBG model by merging it with our sequence generation model.

5.5.1 PRNG based on Integer sequences

Following [DPR+13; WS19], a PRNG with input is a triplet of deterministic polynomial-time algorithms ($instantiate, generate, reseed$). Utilizing the distinctive properties of $\mathcal{M}_{IS}^{\mathbb{N}}$, which are comprehensively detailed in Figure 4.1, we capitalize on its unique features to develop our PRNG. Specifically, we configure $\mathcal{M}_{IS}^{\mathbb{N}}$ to function effectively as a block cipher. The PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$. is initialised by invoking the instantiate function with an entropy sample I and a nonce N, producing the initial state S_0. Subsequently, the generate function accepts a state S, the desired number of output bits $nbits$, an additional input known as an $addin$, and yields a new state S' alongside the bits $R \in \{0,1\}^{\text{nbits}}$. Lastly, the reseed function takes a state S, an entropy sample I, and an extra input $addin$, resulting in a new state S'.

We instead assume that the PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ correctly receives entropy samples drawn uniformly at random from the entropy space, which better matches our real-world scenario.

Private State. The private state S of the PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ is composed of the following:

– A key $K \in \{0,1\}^{\text{keylen}}$, with bit length keylen that matches that of the underlying cipher.

– A counter $V \in \{0,1\}^{\leq\text{blocklen}}$ that increments after each call to the block cipher, where blocklen is the output length of the underlying block cipher.

– A reseed counter c that indicates when a reseed is required. The PRNG's nonce space N is $\{0,1\}^{\text{seedlen}}$ and the entropy space is $\{0,1\}^{\text{seedlen}}$ where the $seedlen = keylen + blocklen$.

Figure 5.3 illustrates the flow chart for the PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ algorithm. PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ functions as a deterministic random number generator, designed to preserve and verify the integrity of inputs directed to the output-generation function. This function implements a deterministic algorithm to protect the entropy properties from external threats. As illustrated in Figure 5.3, the random number generator commences by establishing the initial internal state, which is then periodically updated, and this updated state is utilized to produce output (random numbers). The process of defining the initial internal state is called the instance creation (initialization) function, whereas the process that modifies the internal state is referred to as the internal state update function. The process responsible for generating the output is identified as the output generation function.

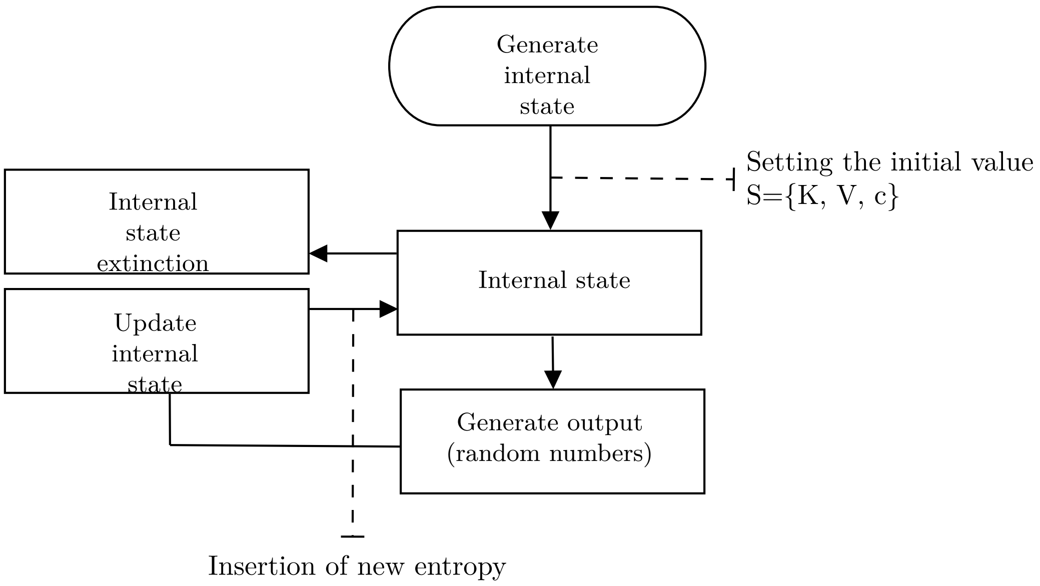

Generate internal state
Setting the initial value S={K, V, c}
Internal state extinction
Internal state
Update internal state
Generate output (random numbers)
Insertion of new entropy

Figure 5.3 Flowchart of PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$.

The critical values for the internal state, on which the security of this PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ mechanism relies, are represented by V and K. Specifically, V and K are the "secret values" that constitute the internal state.

Instantiation. PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ instantiate function takes as input an entropy sample I and an arbitrary nonce N chosen by the implementation, of equal length. It computes a temporary value t as the output for the derivation function applied to I and N. It then calls a subroutine update, outlined in Algorithm 4, with inputs $K = V = 0$ and t as the additional input. The initial state $S_0 = (K, V, c)$ consists of the outputs (K, V) from update, and reseed counter $c = 1$.
Update internal state. Each of PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$'s functions call a subroutine update, outlined in Algorithm 4, that updates the internal state. The routine's input is a key K, counter V, and additional data *addin*. In Lines 3-5 the function increments the counter V and appends the encryption of V under key K to a buffer temp. This process is repeated until temp contains *seedlen* bytes. The resulting buffer is then XORed with *addin* (Line 7). Finally, in Lines 8-9 the function outputs the new key K' as the leftmost *keylen* bits of the buffer, and new counter value V' as the rightmost blocklen bits of the buffer, where *blocklen* is the block length of the cipher.

Algorithm 4: PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ Update function.

```
   Input: $V, K, I$
   Output: $V', K'$
1  temp ← null;
2  while $len\ (temp) < seedlen$ do
3      $V \leftarrow (V + 1) \bmod 2^{\mathrm{blocklen}}$ ;
4      outputblock ← encrypt $(K, V)$;
5      temp ← temp || outputblock;
6  end
7  temp ← temp ⊕ addin;
8  $K'$ ← leftmost (temp, keylen);
9  $V'$ ← rightmost(temp, blocklen);
10 return $K', V'$
```

Generate output. A user generates the output of PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ by calling the generate function outlined in Algorithm 5. It takes as input the state S, the number of bits requested *nbits*, and a string addin, and outputs a string nbits in length and an updated state S'. According to [RBG15], the *addin* parameter "may be a means of providing more entropy for the DRBG internal state". This additional input is allowed to be public or private and may contain secrets if private. The specification notes that "if the additional input is kept secret and has sufficient entropy, the input can provide more assurance when recovering from the compromise of the entropy input, the seed or one or more DRBG internal states". However, the specification does not include requirements for secrecy or entropy for *addin*.
The generate function first checks if a reseed is needed and, if so, throws an error (Lines 2-4).

If the call included additional data *addin*, these data are first whitened by running them through the derivation function, and then used to update K and V through a call to update (Lines 5-8). Otherwise, *addin* is set to a string of zeros (Line 9-11). On each iteration of the loop on Lines 13-17, the counter V is incremented. V is then encrypted under K and the result is appended to the output buffer. This

Algorithm 5: PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ Generate function

```
   Input: $S$, nbits, addin
   Output: $S = (K', V', c')$, out
 1 parse $(K, V, c)$ from $S$;
 2 if $c >$ reseed-interval then
 3     return reseed-required
 4 end
 5 if addin $\neq$ Null then
 6     addin $\leftarrow$ df(addin);
 7     $(K, V) \leftarrow$ update $(K, V,$ addin);
 8 end
 9 else
10     addin $\leftarrow 0^{\text{seedlen}}$ ;
11 end
12 temp $\leftarrow$ Null;
13 while len (temp) $<$ nbits do
14     $V \leftarrow (V + 1) \bmod 2^{\text{blocklen}}$ ;
15     output-block $\leftarrow$ encrypt$(K, V)$;
16     temp $\leftarrow$ temp $\|$ output-block;
17 end
18 out $\leftarrow$ leftmost(temp, nbits);
19 $(K', V') \leftarrow$ update(addin, $K, V$);
20 $c' \leftarrow c + 1$;
21 return $S = (K', V', c')$, out
```

process is repeated until enough output has been collected. On Line 19 the function calls the update with *addin* to update K and V again before the reseed counter c is incremented (Line 20). The function returns the new state S and output.

If the attacker compromises the key K and counter V between Lines 13-17 and is able to guess *addin*, she can predict the new key K' and counter V'. She can then predict future PRNG outputs as well as future values of K and V. Note that the same symmetric key is used to generate all of the requested output, and the key is only changed at Line 19 after all blocks have been generated. This observation is a crucial element of our attack, as a long output buffer gives the attacker many opportunities to extract K via a side channel. In fact, [RBG15] specifies that at most 65 KB can be requested from the generator in a single call before a key change. This is presumably intended to limit the exposure of a single state to an attacker. However, our work demonstrates that state recovery attacks within this limit are still viable.

Reseeding. The reseed function is intended to ensure that high-quality entropy is mixed into the state as required. The reseed function takes as input an additional input *addin*, an entropy sample I, and a state S consisting of the key K, counter V, and the reseed counter c. It calls the update subroutine on a derivation function taken over I and *addin*, which updates K and V. Finally, it resets the reseed counter c to 1 and returns the new key, counter, and reseed counter.

Algorithm 6: PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ Reseed function.

```
  Input: $S = (V, K, c), I$
  Output: $S' = (V', K', c')$
1 $(K', V') = $ PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ Update $(V, K, I)$;
2 $c' = 1$;
3 return $(V', K', c')$
```

5.5.2 Statistical tests on PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$

The randomness of the bit sequences produced by the proposed generator has been evaluated using three different statistical software packages described in Section 5.4.4.

(**NIST STS**). We provided a dataset comprising 1000 sequences, each consisting of 1,000,000 bits, to evaluate using the NIST statistical tests. The results are detailed in Table 5.4.

The complete NIST test is successfully passed: all p-values across all 1000 sequences exhibit a uniform distribution over the 10 subintervals, with the pass rate falling within an acceptable range. For each statistical test, excluding the random excursion (variant) test, the minimum pass rate is approximately 980. For the random excursion (variant) test, the minimum pass rate is roughly 596 based on a sample of 610 binary sequences.

(**Diehard**). In the Diehard tests, we produced a file containing 80 million bits. The outcomes are presented in Table 5.5.

NIST statistical test	P-value	Pass rate
Frequency (monobit)	0.078132	986/1000
Block-frequency	0.482232	987/1000
Cumulative sums (Forward)	0.694587	987/1000
Cumulative sums (Reverse)	0.749847	987/1000
Runs	0.799571	990/1000
Longest run of Ones	0.083526	987/1000
Rank	0.894835	990/1000
FFT (Spectral)	0.008598	984/1000
Non-overlapping templates	0.534926	989/1000
Overlapping templates	0.549818	987/1000
Universal	0.603816	986/1000
Approximate entropy	0.492371	989/1000
Random-excursions	0.472934	602/610
Random-excursions Variant	0.548874	604/610
Serial 1	0.177362	991/1000
Serial 2	0.171726	990/1000
Linear complexity	0.468326	992/1000

Table 5.4 NIST Statistical test suite results for 1000 sequences of size 1 million bits each generated by PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$.

DIEHARD statistical tests	P-value
Birthday spacings	0.493827
Overlapping 5-permutation	0.384735
Binary rank (31×31)	0.522837
Binary rank (32×32)	0.883712
Binary rank (6×8)	0.334985
Bitstream	0.589842
OPSO	0.379583
OQSO	0.534094
DNA	0.493824
Stream count-the-ones	0.370932
Byte count-the-ones	0.572421
Parking lot	0.402842
Minimum distance	0.510924
3D spheres	0.482756
Squeeze	0.462439
Overlapping sums	0.563958
Runs up	0.172834
Runs down	0.845920
Craps	0.593847

Table 5.5 Diehard statistical test results for 80 million bits

Every p-value falls within the valid interval of [0,1). The suggested PRNG successfully passed all tests in the Diehard suite.
(**ENT**). We evaluated an output string consisting of 125,000,000 bytes from PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$. The findings are detailed in Table 5.6.

ENT statistical tests	Results
Entropy	7.999999 bits per byte
Optimum compression	OC would reduce the size of this 125000000 byte file by 0%.
χ^2 distribution	For 125000000 samples is 254.38 , and randomly would exceed this value 49.92% of the times.
Arithmetic mean value	127.5021(127.5 = random)
Monte Carlo π estimation	3.141740786 (error 0.00%)
Serial correlation coefficient	-0.000221 (totally uncorrelated =0.0)

Table 5.6 ENT statistical test results for 125000000 bytes.

The output stream was successfully assessed and passed all rigorous evaluations and assessments performed using the ENT package.

5.5.3 Performance tests on PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$

After to conducting a comprehensive statistical examination to ascertain the degree of stochasticity inherent in PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$, we advanced to assess various performance metrics by implementing the tests on a general-use personal computing system. The computational unit employed in this study is an Intel® Core™ i7-1370PRE Processor, belonging to the 13th generation, operating at a frequency of 4.80 GHz. The code compilation was performed using version 13.1 of the GNU Compiler Collection (GCC). The GNU Compiler Collection (GCC) represents a comprehensive suite of compiler tools tailored for a multitude of programming languages. This collection encompasses compilers specifically designed for languages such as C, C++, Fortran, Ada, among others. Distinguished as an open-source initiative, GCC is widely adopted in the domain of software development. Its wide usage can be attributed to its cost-free availability, exceptional portability, and compatibility with an extensive array of computer architectures.

In order to evaluate performance, we compared our results with the baseline, specifically focussing on the slowest PRNG available, which is characterised by the base implementation of the rand function (Figure 5.4). The execution time measurements were conducted using 10^8 randomly generated numbers, which were processed on our Processor (Figure 5.5).

Regarding the experimental evaluations that were carried out, it is pivotal to delineate the subsequent clarifications:

– The rate of processing can vary significantly between various CPUs and in various operational contexts.

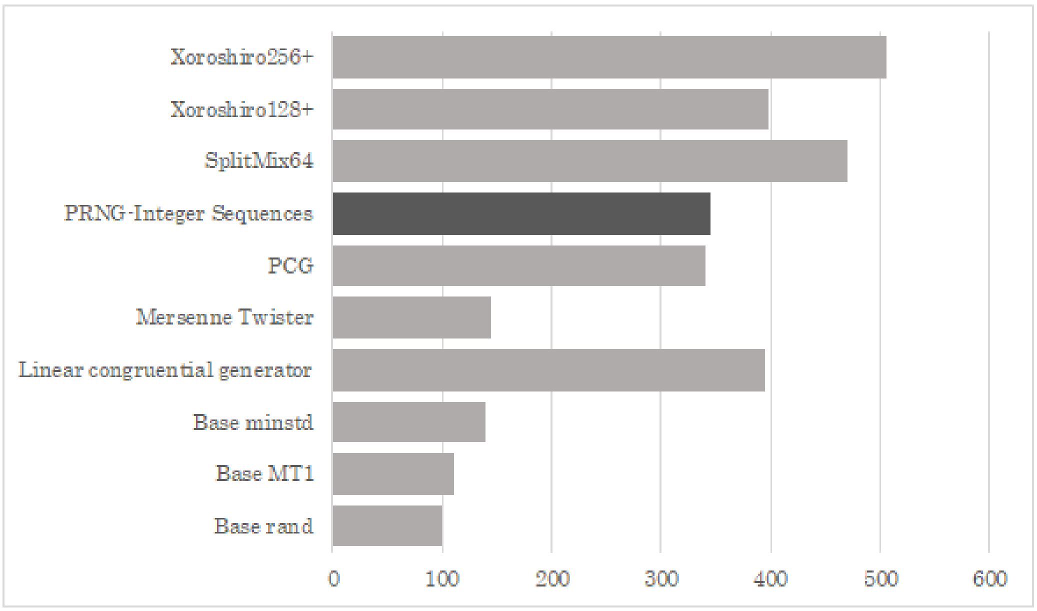


Figure 5.4 PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ performance (%) compared (a high value indicates a better result).

.

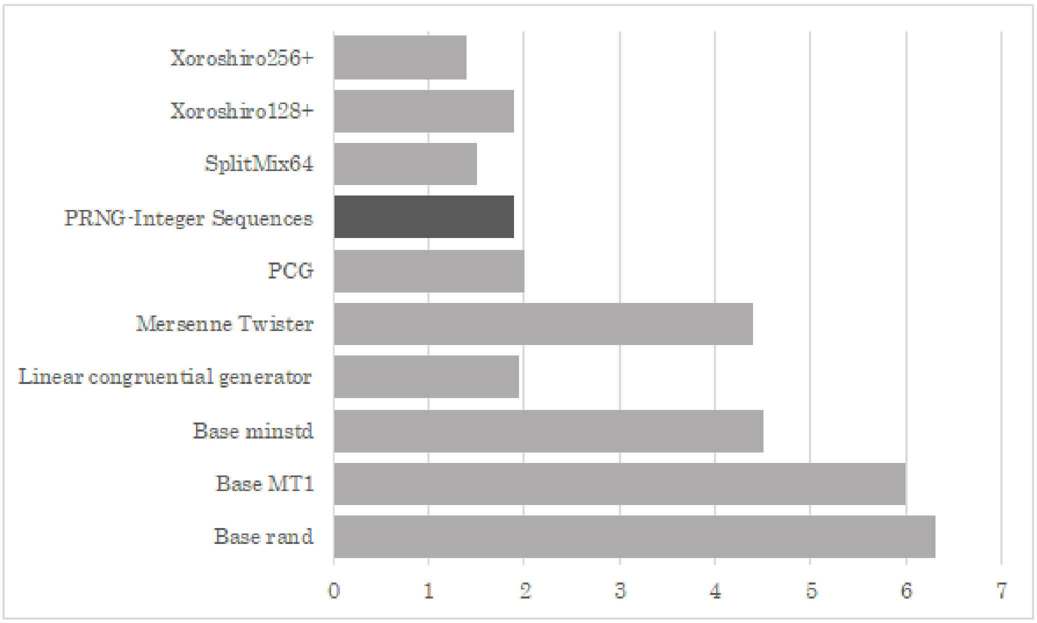


Figure 5.5 PRNG-$\mathcal{M}_{IS}^{\mathbb{N}}$ time (s) compared (lower value indicates a better result).

.

– We abstain from using the `-march=native` compiler flag, which could potentially enhance the execution time of certain generators through the application of vectorisation or other architecture-specific instructions. However, we refrain from this practice because such improvements may not be achievable when the generator is integrated into the user-level code. The `-march=native` option is a flag used in the GCC compiler (and other GCC-based compilers) to optimise the code for the specific architecture of the machine on which the programme is being compiled. When using `-march=native`, GCC automatically detects the architecture of the CPU on the system where the programme is being compiled (based on information provided by the operating system) and optimises the generated code for that specific architecture. This can include using special instructions, advanced registers, and other processor-specific optimisations to improve the programme's performance.

– Instead of being a typical method for employing generators, timing measurements were performed by executing the generator billions of times in a loop. When such generators are integrated within a software application, the allocation of registers may vary significantly. This may lead to scenarios where constants need to be reloaded or portions of the state space are written to the main memory during each iteration. It is crucial to note that these particular costs are not represented in the following benchmarks.

– The code has been compiled using GCC with the option `-fno-unroll-loops`. This particular option is crucial for obtaining meaningful results. When it is not present, the compiler may apply varying strategies for loop unrolling, depending on the generator employed. Previously, we also used the flag `-fno-move-loop-invariants` as it was crucial in ensuring fairness for generators that rely on multiple large constants. This flag prevents the compiler from prematurely loading these constants into registers, thereby avoiding any potential advantage.

Chapter 6

Conclusions and problems

The journey undertaken in this doctoral research has been a unique opportunity to explore and deepen the understanding of the use of integer sequences in cryptography, with the goal of making a meaningful contribution in this field.

The key outcomes of this study are:

1. For the first time, a systematic literature study has been conducted that highlighted the importance of using integer sequences in cryptography. The sequences that have made the most substantial impact on various cryptographic sub-domains have been determined. This enabled the establishment of a structured framework for collecting individual contributions. By applying this framework, it became possible to ascertain which specific sub-domains within cryptography have most extensively utilized integer sequences for their applications.

2. The discovery of a Model capable of identifying a new family of integer sequences, most of which are unknown in the On-Line Encyclopedia of Integer Sequences (OEIS). This family apparently is capable of also encompassing every sequence of prime numbers and can represent a significant result in the field of Number Theory. Furthermore, as hypothesized, the Model can be further developed by highlighting new mathematical properties that are not yet documented in the literature.

3. The application of the aforementioned Model in the field of cryptography. In particular, for defining a Pseudo-Random Number Generator (PRNG) with good randomness properties that emerged from statistical tests and endowed with adequate performance compared to the most common PRNGs.

Although this research has achieved its main objectives, it also opens the door to future work. It is my belief that the outcomes of this research, coupled with continued investigation, will lead to further progress and understanding in use of integer sequences in cryptography. Finally, it has been a privilege to contribute to this field, and we look forward to seeing how future scholars and practitioners build upon the work presented here.

Appendix - Examples of generable integer sequences

n	F_n	$F_n^{ST[w]}$	S_m	$S_m \subset \mathcal{P}$	**OEIS check**	**OEIS Info**
500	[1, 2, 0, -2, 0, 2, 0, -2, 0, 2, 0, -2, 0, 1, 0, -2, 0, 1, 0, -2, 0, 1, 0, -2, 0, . . .]	[0, 0, 0, 2, 0, 2, 0, 6, 0, 2, 0, 10, 0, 1, 0, 14, 0, 1, 0, 18, 0, 1, 0, 22, 0, . . .]	[3, 5, 7, 11, 13, 17, 31, 41, 47, 59, 61, 83, 97, 103, 137, 149, 151, 173, 193, 227, 229, 233, 241, 283, 293, 311, 373, 389, 401, 409, 457, ...]	Yes	Unknow	
500	[1, 4, 0, -4, 0, 4, 0, -4, 0, 4, 0, -4, 0, 3, 0, -4, 0, 3, 0, -4, 0, 3, 0, -4, 0, . . .]	[0, 0, 0, 0, 0, 4, 0, 4, 0, 4, 0, 8, 0, 3, 0, 12, 0, 3, 0, 16, 0, 3, 0, 20, 0, . . .]	[3, 5, 7, 11, 13, 19, 23, 37, 41, 43, 61, 67, 71, 79, 83, 89, 97, 103, 127, 131, 137, 149, 151, 163, 167, 173, 191, 233, 239, 251, 271, 281, 307, 313, 331, 337, 347, 401, 439, 449, 463, 499, ...]	Yes	Unknow	
500	[1, 6, 0, -6, 0, 6, 0, -6, 0, 6, 0, -6, 0, 5, 0, -6, 0, 5, 0, -6, 0, 5, 0, -6, 0, . . .]	[0, 0, 0, 2, 0, 0, 0, 2, 0, 6, 0, 6, 0, 5, 0, 10, 0, 5, 0, 14, 0, 5, 0, 18, 0, . . .]	[3, 5, 7, 11, 13, 17, 29, 31, 37, 43, 67, 73, 79, 83, 97, 109, 151, 157, 179, 191, 193, 197, 199, 223, 233, 241, 263, 307, 337, 359, 367, 373, 389, 409, 433, 457, 463, 487, ...]	Yes	Unknow	

Continued on next page

Table A1 – continued from previous page

n	F_n	$F_n^{ST[w]}$	S_m	$S_m \subset \mathcal{P}$	**OEIS check**	**OEIS Info**
500	[3, 0, -3, -6, -1, -4, -7, -10, -5, -8, -11, -14, -9, -12, -15, -18, -13, -16, ...]	[0, 0, 0, 2, 4, 2, 0, 6, 4, 2, 0, 10, 4, 2, 0, 14, 4, 2, 0, 18, 4, 2, 0, 22, 4, ...]	[3, 5, 7, 11, 23, 47, 71, 83, 107, 131, 167, 227, 311, 383, 443, 467, 491, ...]	Yes	Unknow	
500	[3, 2, -3, -4, -1, -2, -7, -8, -5, -6, -11, -12, -9, -10, -15, -16, -13, -14, ...]	[0, 0, 0, 0, 4, 4, 0, 0, 4, 4, 0, 0, 4, 4, 0, 0, 4, 4, 0, 0, 4, 4, 0, 0, 4, 4, 0, ...]	[3, 5, 7, 11, 23, 47, 71, 83, 107, 131, 167, 227, 311, 383, 443, 467, 491, ...]	Yes	Unknow	
500	[4, 2, -6, -26, -56, -118, -210, -338, -500, -718, -990, -1322, -1712, ...]	[0, 0, 0, 2, 4, 2, 0, 6, 4, 2, 0, 10, 4, 2, 0, 14, 4, 2, 0, 18, 4, 2, 0, 22, 4, ...]	[3, 5, 7, 11, 23, 47, 71, 83, 107, 131, 167, 227, 311, 383, 443, 467, 491, ...]	Yes	Unknow	
500	[4, 6, 6, 12, 24, 34, 42, 56, 76, 94, 110, 132, 160, 186, 210, 240, ...]	[0, 0, 0, 0, 4, 4, 0, 0, 4, 4, 0, 0, 4, 4, 0, 0, 4, 4, 0, 0, 4, 4, 0, ...]	[3, 5, 7, 11, 23, 47, 71, 83, 107, 131, 167, 227, 311, 383, 443, 467, 491, ...]	Yes	Unknow	
500	[4, 2, 0, -2, 4, 2, 0, -2, 4, 2, 0, -2, 4, 2, 0, -2, 4, 2, 0, -2, 4, 2, 0, -2, 4, ...]	[0, 0, 0, 2, 4, 2, 0, 6, 4, 2, 0, 10, 4, 2, 0, 14, 4, 2, 0, 18, 4, 2, 0, 22, 4, ...]	[3, 5, 7, 11, 23, 47, 71, 83, 107, 131, 167, 227, 311, 383, 443, 467, 491, ...]	Yes	Unknow	
500	[8, 4, 0, -4, 8, 4, 0, -4, 8, 4, 0, -4, 8, 4, 0, -4, 8, 4, 0, -4, 8, 4, 0, -4, 8, ...]	[0, 0, 0, 0, 3, 4, 0, 4, 8, 4, 0, 8, 8, 4, 0, 12, 8, 4, 0, 16, 8, 4, 0, 20, 8, ...]	[3, 5, 7, 11, 19, 31, 67, 79, 139, 199, 271, 367, 439, 487, 499, ...]	Yes	Unknow	
500	[12, 6, 0, -6, 12, 6, 0, -6, 12, 6, 0, -6, 12, 6, 0, -6, 12, 6, 0, -6, 12, 6, 0, ...]	[0, 0, 0, 2, 2, 0, 0, 2, 3, 6, 0, 6, 12, 6, 0, 10, 12, 6, 0, 14, 12, 6, 0, 18, ...]	[3, 5, 11, 13, 19, 23, 31, 43, 59, 71, 79, 83, 139, 151, 163, 179, 191, 211, 223, 239, 251, 263, 283, 359, 379, 431, 443, 479, 491, 499, ...]	Yes	Unknow	
500	[16, 8, 0, -8, 16, 8, 0, -8, 16, 8, 0, -8, 16, 8, 0, -8, 16, 8, 0, ...]	[0, 0, 0, 0, 1, 2, 0, 0, 7, 8, 0, 4, 3, 8, 0, 8, 16, 8, 0, 12, 16, 8, 0, 16, ...]	[3, 5, 7, 13, 17, 19, 23, 47, 59, 83, 167, 179, 227, 239, 347, 383, 479, ...]	Yes	Unknow	

Continued on next page

Table A1 – continued from previous page

n	F_n	$F_n^{ST[w]}$	S_m	$S_m \subset \mathcal{P}$	**OEIS check**	**OEIS Info**
500	[20, 10, 0, -10, 20, 10, 0, -10, 20, 10, 0, -10, 20, 10, 0, -10, 20, 10, 0, . . .]	[0, 0, 0, 2, 0, 4, 0, 6, 2, 0, 0, 2, 7, 10, 0, 6, 3, 10, 0, 10, 20, 10, 0, 14, . . .]	[3, 5, 7, 13, 17, 19, 23, 31, 43, 67, 79, 103, 127, 151, 199, 211, 271, 283, 331, 367, 379, 439, 463, 487, 499, ...]	Yes	Unknow	
500	[3, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, . . .]	[0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, . . .]	[3, 7, 23, 47, 167, 263, 359, 383, 479, ...]	Yes	A158035	2 * A158034 + 1, prime numbers p for which $f = (2^p - 2^{(p-1)}/2 + 1) + 4p^2 - 8p)/(2p^2 - 2p)$ *is an integer*
600	[0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, . . .]	[0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, . . .]	[3, 11, 59, 83, 107, 131, 179, 227, 251, 347, 443, 467, ...]	Yes	A199854	Primes of the form 1 + m^2 + n^2 with gcd(m,n)=1.
1000	[1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, . . .]	[0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, . . .]	[3, 5, 11, 53, 59, 107, 149, 347, 587, ...]	Yes	Unknow	
500	[0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, . . .]	[0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, . . .]	[3, 83, 179, 227, 467, ...]	Yes	Unknow	
500	[0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, . . .]	[0, 1, 0, 3, 0, 1, 0, 7, 0, 1, 0, 11, 0, 1, 0, 15, 0, 1, 0, 19, 0, 1, 0, 23, 0, . . .]	[4, 8, 16, 32, 64, 128, 256, ...]	No	A000079	Powers of 2: a(n) = 2^n.
500	[1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, . . .]	[0, 0, 2, 0, 1, 0, 6, 0, 1, 0, 10, 0, 1, 0, 14, 0, 1, 0, 18, 0, 1, 0, 22, 0, 1, . . .]	[3, 7, 9, 15, 25, 33, 39, 49, 57, 63, 129, 135, 159, 177, 193, 207, 225, 249, 255, 327, 345, 369, 385, 399, 423, ...]	No	Unknow	

Continued on next page

Table A1 – continued from previous page

n	F_n	$F_n^{ST[w]}$	S_m	$S_m \subset \mathcal{P}$	**OEIS check**	**OEIS Info**
500	[1, 0, 0, -1, 0, 0, 1, 0, 0, -1, 0, 0, 1, 0, 0, -1, 0, 0, 1, 0, 0, -1, 0, 0, 1, 0 ...]	[0, 0, 0, 3, 0, 0, 1, 0, 0, 9, 0, 0, 1, 0, 0, 15, 0, 0, 1, 0, 0, 21, 0, 0, 1, 0, ...]	[3, 5, 13, 17, 37, 41, 61, 73, 97, 133, 181, 193, 217, 233, 257, 277, 361, 397, 433, 481, ...]	No	Unknow	
500	[-1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, ...]	[0, 0, 1, 0, 4, 0, 1, 0, 8, 0, 1, 0, 12, 0, 1, 0, 16, 0, 1, 0, 20, 0, 1, 0, 24, ...]	[3, 5, 11, 21, 27, 75, 117, 123, 165, 171, 213, 261, 315, 357, 411, 453, ...]	No	Unknow	
500	[1, 3, 0, -3, 0, 3, 0, -3, 0, 3, 0, -3, 0, 2, 0, -3, 0, 2, 0, -3, 0, 2, 0, -3, 0, ...]	[0, 1, 0, 1, 0, 3, 0, 5, 0, 3, 0, 9, 0, 2, 0, 13, 0, 2, 0, 17, 0, 2, 0, 21, 0, ...]	[4, 8, 32, 64, 172, 424, ...]	No	Unknow	
1000	[1, 5, 0, -5, 0, 5, 0, -5, 0, 5, 0, -5, 0, 4, 0, -5, 0, 4, 0, -5, 0, 4, 0, -5, 0, ...]	[0, 1, 0, 3, 0, 5, 0, 3, 0, 5, 0, 7, 0, 4, 0, 11, 0, 4, 0, 15, 0, 4, 0, 19, 0, ...]	[4, 8, 16, 64, 548, ...]	No	Unknow	
500	[2, 3, 0, 1, 6, 7, 4, 5, 10, 11, 8, 9, 14, 15, 12, 13, 18, 19, 16, 17, 22, ...]	[0, 1, 0, 1, 1, 1, 4, 5, 1, 1, 8, 9, 1, 1, 12, 13, 1, 1, 16, 17, 1, 1, 20, 21, ...]	[4, 8, 20, 44, 68, 80, 104, 128, 164, 224, 308, 380, 440, 464, 488, ...]	No	Unknow	
500	[3, 2, 1, 0, 7, 6, 5, 4, 11, 10, 9, 8, 15, 14, 13, 12, 19, 18, 17, 16, 23, ...]	[0, 0, 1, 0, 2, 0, 5, 4, 2, 0, 9, 8, 2, 0, 13, 12, 2, 0, 17, 16, 2, 0, 21, 20, ...]	[3, 5, 9, 21, 45, 69, 81, 105, 129, 165, 225, 309, 381, 441, 465, 489, ...]	No	Unknow	
1200	[1, 3, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, ...]	[0, 1, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, ...]	[6, 18, 66, 258, 1026, ...]	No	A178789	$4^{(n-1)} + 2$: Number of acute angles after n iterations of the Koch snowflake construction.

Table A1 Several instances of sequences produced by the model $\mathcal{M}_{IS}^{\mathbb{N}}$

n	$F_n^{ST[w]}$	S_m	$S_m \subset \mathcal{P}$	**OEIS check**	**OEIS Info**
44	[0, 0, 0, 2, 0, 2, 4, 4, 6, 7, 3, 6, 2, 9, 3, 0, 15, 10, 18, 15, 11, 12, 21, 6, 24, 20, 2, 12, 13, 4, 21, 14, 30, 12, 22, 12, 2, 11, 31, 9, 29, 33, 25, 32]	[3, 5, 7, 13, 19, 31, 43, ...]	Yes	A006512	Greater of twin primes.
42	[0, 0, 1, 0, 3, 3, 0, 2, 0, 5, 5, 3, 1, 10, 1, 7, 13, 7, 15, 8, 18, 4, 9, 22, 8, 2, 24, 18, 11, 17, 16, 0, 9, 24, 3, 1, 26, 31, 33, 17, 22, 37]	[3, 5, 11, 17, 29, 41...]	Yes	A077800	List of twin primes p, p+2.
54	[0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[3, 5, 11, 23, 29, 41, 53...]	Yes	A005384	Sophie Germain primes p: 2p+1 is also prime.

Table A2 Examples of well-known Prime number sequences extracted from the model $\mathcal{M}_{IS}^{\mathbb{N}}$

Bibliography

[AALA16] Nur Azman Abu, Shekh Faisal Abdul-Latip, and Muhammad Rezal Kamel Ariffin. A comparative s-index in factoring rsa modulus via lucas sequences. *Cryptology ePrint Archive*, 2016. URL: https://eprint.iacr.org/2016/937.

[AB74] R. C. Agarwal and C. Burrus. Fast convolution using fermat number transforms with applications to digital filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 22(2):87–97, 1974. doi:10.1109/TASSP.1974.1162555.

[AB75] R. C. Agarwal and C. S. Burrus. Number theoretic transforms to implement fast digital convolution. *Proceedings of the IEEE*, 63(4):550–560, 1975. doi:10.1109/PROC.1975.9791.

[ACS06] Sos S. Agaian, Ravindranath C. Cherukuri, and Ronnie Sifuentes. A new secure adaptive steganographic algorithm using fibonacci numbers. In *2006 IEEE Region 5 Conference*, pages 125–129, 2006. doi:10.1109/TPSD.2006.5507446.

[AF87] Alberto Apostolico and Aviezri S. Fraenkel. Robust transmission of unbounded strings using fibonacci representations. *IEEE Transactions on Information Theory*, 33(2):238–245, 1987. doi:10.1109/TIT.1987.1057284.

[AGB10] Mohammed Falih Al-Gailani and Said Boussakta. Evaluation of one-dimensional nmnt for security applications. In *2010 7th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP 2010)*, pages 715–720. IEEE, 2010. doi:10.1109/CSNDSP16145.2010.5580331.

[AJ96] Jean-Paul Allouche and Tom Johnson. Narayana's cows and delayed morphisms. In *Journées d'Informatique Musicale*, île de Tatihou, France, May 1996. URL: https://hal.science/hal-02986050.

[AJPS18] Divesh Aggarwal, Antoine Joux, Anupam Prakash, and Miklos Santha. A new public-key cryptosystem via mersenne numbers. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part III 38*, pages 459–482. Springer, 2018. doi:10.1007/978-3-319-96878-0_16.

[AKH13] F. Amounas, E. Kinani, and M. Hajar. A novel approach for enciphering data based ecc using catalan numbers. *International Journal of Information and Network Security (IJINS)*, 24, 2013. doi:10.11591/IJINS.V2I4.3447.

[Ala10] Mohammed M Alani. Testing randomness in ciphertext of block-ciphers using diehard tests. *Int. J. Comput. Sci. Netw. Secur*, 10(4):53–57, 2010. URL: http://paper.ijcsns.org/07_book/201004/20100409.pdf.

[Alu97] Srinivas Aluru. Lagged fibonacci random number generators for distributed memory parallel computers. *Journal of Parallel and Distributed Computing*, 45(1):1–12, 1997. doi:10.1006/jpdc.1997.1363.

[And95] Ross Anderson. On fibonacci keystream generators. In *Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14-16, 1994 Proceedings 2*, pages 346–352. Springer, 1995. doi:10.1007/3-540-60590-8_26.

[AT21] Talha Burak Alakuş and İbrahim Türkoğlu. A novel fibonacci hash method for protein family identification by usingrecurrent neural networks. *Turkish Journal of Electrical Engineering and Computer Sciences*, 29(1):370–386, 2021. doi:10.3906/elk-2003-116.

[Bar05] Paul Barry. A catalan transform and related transformations on integer sequences. *Journal of Integer Sequences*, 8, 2005. URL: https://cs.uwaterloo.ca/journals/JIS/VOL8/Barry/barry84.pdf.

[BBL95] Daniel Bleichenbacher, Wieb Bosma, and Arjen K. Lenstra. Some remarks on lucas-based cryptosystems. In *Annual International Cryptology Conference*, pages 386–396. Springer, 1995. doi:10.1007/3-540-44750-4_31.

[BD00] Dan Boneh and Glenn Durfee. Cryptanalysis of rsa with private key d less than n/sup 0.292/. *IEEE Transactions on Information Theory*, 46(4):1339–1349, Jul 2000. doi:10.1109/18.850673.

[BDG20] Jhon J. Bravo, Pranabesh Das, and Sergio Guzmán. Repdigits in narayana's cows sequence and their consequences. *Journal of Integer Sequences*, 23(8):1–15, 2020. URL: https://cs.uwaterloo.ca/journals/JIS/VOL23/Das/bravo17.pdf.

[BEB06] Robert G. Brown, Dirk Eddelbuettel, and David Bauer. Dieharder: A random number test suite, 2006. URL: https://webhome.phy.duke.edu/~rgb/General/dieharder.php.

[BH95] S. Boussakta and A.G.J. Holt. New transform using the mersenne numbers. *IEE Proceedings-Vision, Image and Signal Processing*, 142(6):381–388, 1995. doi:10.1049/ip-vis:19952323.

[BLS75] John Brillhart, Derrick H. Lehmer, and John L. Selfridge. New primality criteria and factorizations of $2^m \pm 1$. *Mathematics of computation*, 29(130):620–647, 1975. doi:10.1090/s0025-5718-1975-0384673-1.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984. doi:10.1137/0213053.

[BMM21] Emanuele Bellini, Chiara Marcolla, and Nadir Murru. An application of p-fibonacci error-correcting codes to cryptography. *Mathematics*, 9(7):789, 2021. doi:10.3390/math9070789.

[Bon99] Dan Boneh. Twenty years of attacks on the rsa cryptosystem. *Notices of the American Mathematical Society*, 46:203–212, 1999. URL: https://www.ams.org/notices/199902/boneh.pdf.

[Boy89a] Joan Boyar. Inferring sequences produced by a linear congruential generator missing low-order bits. *Journal of Cryptology*, 1(3):177–184, 1989. doi:10.1007/BF02252875.

[Boy89b] Joan Boyar. Inferring sequences produced by pseudo-random number generators. *J. ACM*, 36(1):129–141, jan 1989. doi:10.1145/58562.59305.

[Bro14] Daniel R.L. Brown. Cm55: special prime-field elliptic curves almost optimizing den boer's reduction between diffie-hellman and discrete logs. 2014. URL: https://eprint.iacr.org/2014/877.

[Bru15] Viggo Brun. Uber das goldbachsche gesetz und die anzahl der primzahlpaare. *Arch. Mat. Natur. B*, 1915.

[Bru12] Richard A. Brualdi. *Introductory Combinatorics*. Pearson Education. Pearson Prentice Hall, 2012.

[BV98] Dan Boneh and Ramarathnam Venkatesan. Breaking rsa may not be equivalent to factoring. In *International Conference on the Theory and Applic ation of Cryptographic Techniques*, 1998. doi:10.1007/bfb0054117.

[BW80] Robert Baillie and Samuel S. Wagstaff. Lucas pseudoprimes. *Mathematics of Computation*, 35(152):1391–1417, 1980. doi:10.1090/S0025-5718-1980-0583518-6.

[BW23] Christian J.C. Ballot and Hugh C Williams. *The Lucas Sequences: Theory and Applications*, volume 8. Springer Nature, 2023. doi:10.1007/978-3-031-37238-4.

[Cae92] William Caelli. Crypt-x package documentation. information security research centre and school of mathematics, 1992.

[Cap90] Renato M. Capocelli. Flag encodings related to the zeckendorf representation of integers. In *Sequences: Combinatorics, Compression, Security, and Transmission*, pages 449–466. Springer, 1990. doi:10.1007/978-1-4612-3352-7_36.

[Cas07] Guilhem Castagnos. An efficient probabilistic public-key cryptosystem over quadratic fields quotients. *Finite Fields and Their Applications*, 13(3):563–576, 2007. doi:10.1016/j.ffa.2006.05.004.

[CEF19] Abdelhakim Chillali and Lhoussain El Fadil. Anonymous multi-receiver public key encryption based on third order linear sequences. In *AIP Conference Proceedings*, volume 2074. AIP Publishing, 2019. doi:10.1063/1.5090630.

[Che06] Jung Hee Cheon. Security analysis of the strong diffie-hellman problem. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–11. Springer, 2006. doi:10.1007/11761679_1.

[CM05] Alina Carmen Cojocaru and M Ram Murty. *An introduction to sieve methods and their applications*. Number 66. Cambridge University Press, 2005. doi:10.1017/cbo9780511615993.

[CVE08] CVE. Cve-2008-0166. openssl 0.9.8c-1 up to versions before 0.9.8g-9 on debian-based operating systems uses a random number generator that generates predictable numbers, which makes it easier for remote attackers to conduct brute force guessing attacks against cryptographic keys., 2008. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-0166.

[CXM$^+$13] Mei Chen, Qingmei Xiao, Kazuyuki Matsumoto, Minoru Yoshida, Xin Luo, and Kenji Kita. A fast retrieval algorithm based on fibonacci hashing for audio fingerprinting systems. In *2013 International Conference on Advanced Information Engineering and Education Science (ICAIEES 2013)*, pages 219–222. Atlantis Press, 2013. doi:10.2991/icaiees-13.2013.59.

[DH10] C. Doty-Humphrey. Practically random: C++ library of statistical tests for rngs., 2010. URL: https://sourceforge.net/projects/pracrand.

[DPR$^+$13] Yevgeniy Dodis, David Pointcheval, Sylvain Ruhault, Damien Vergniaud, and Daniel Wichs. Security analysis of pseudo-random number generators with input: /dev/random is not robust. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, page 647–658, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2508859.2516653.

[DS19] Monojit Das and Sudipta Sinha. A variant of the narayana coding scheme. *Control and Cybernetics*, 48(3):473 – 484, 2019. URL: https://bibliotekanauki.pl/articles/1839131.pdf.

[Dud69] Underwood Dudley. History of a formula for primes. *American Mathematical Monthly*, 76(1):23–23, Jan 1969. doi:10.2307/2316781.

[EES02] H. Elkamchouchi, K. Elshenawy, and H. Shaban. Extended rsa cryptosystem and digital signature schemes in the domain of gaussian integers. In *The 8th International Conference on Communication Systems, 2002. ICCS 2002.*, volume 1, pages 91–95. IEEE, 2002. doi:10.1109/ICCS.2002.1182444.

[EF12] Lhoussain El Fadil. A public-key cryptosystem based on lucas sequences. *Palestine J. Mathe*, 1(2):148–152, 2012. URL: https://pjm.ppu.edu/sites/default/files/papers/paper_1_2_10.pdf.

[Eli75] Peter Elias. Universal codeword sets and representations of the integers. *IEEE transactions on information theory*, 21(2):194–203, 1975. doi:10.1109/TIT.1975.1055349.

[EVDPS+03] Graham Everest, Alfred Jacobus Van Der Poorten, Igor Shparlinski, Thomas Ward, et al. *Recurrence sequences*, volume 104 of *Mathematical surveys and monographs.* American Mathematical Society, 2003.

[GHH97] George Richard Herbert Greaves, Glyn Harman, and Martin Neil Huxley. *Sieve Methods, Exponential Sums, and their Applications in Number Theory*, volume 237. Cambridge University Press, 1997. doi:10.1017/cbo9780511526091.

[GK02] Mark Goresky and Andrew M. Klapper. Fibonacci and galois representations of feedback-with-carry shift registers. *IEEE Transactions on Information Theory*, 48(11):2826–2836, 2002. doi:10.1109/TIT.2002.804048.

[Gol74] Solomon W. Golomb. A direct interpretation of gandhi's formula. *The American Mathematical Monthly*, 81(7):752–754, 1974. doi:10.1080/00029890.1974.11993659.

[Gol82] Solomon W. Golomb. *Shift register sequences.* Aegean Park Press. Leguna Hills-USA, 1982. doi:10.1142/9361.

[Gol01] Oded Goldreich. *Foundations of cryptography: Volume 1, basic applications*, volume 2. Cambridge university press, 2001. doi:10.1017/CBO9780511546891.

[Gor85] John Gordon. Strong primes are easy to find. In *Advances in Cryptology: Proceedings of EUROCRYPT 84 A Workshop on the Theory and Application of Cryptographic Techniques Paris, France, April 9–11, 1984 3*, pages 216–223. Springer, 1985. doi:10.1007/3-540-39757-4_19.

[Gow17] Shreyank N. Gowda. An intelligent fibonacci approach to image steganography. In *2017 IEEE Region 10 Symposium (TENSYMP)*, pages 1–4, 2017. doi:10.1109/TENCONSpring.2017.8070030.

[Goy18] Taras Goy. On identities with multinomial coefficients for fibonacci-narayana sequence. *Annales Mathematicae et Informaticae*, (49):75–84, 2018. doi:10.33039/ami.2018.09.001.

[Gre13] George Greaves. *Sieves in number theory*, volume 43. Springer Science & Business Media, 2013. doi:10.1007/978-3-662-04658-6.

[Gri06] Ralph P. Grimaldi. *Discrete and Combinatorial Mathematics, 5th ed.* Pearson Education India, 2006.

[Gri12] Ralph P. Grimaldi. *Fibonacci and Catalan Numbers.* Wiley, 2012. Available online. doi:10.1002/9781118159743.

[GW67] R. L. Goodstein and C. P. Wormell. Formulae for primes. *The Mathematical Gazette*, 51(375):35–38, 1967. doi:10.2307/3613607.

[HB14] Mounir T. Hamood and Said Boussakta. Efficient algorithms for computing the new mersenne number transform. *Digital Signal Processing*, 25:280–288, 2014. doi:10.1016/j.dsp.2013.10.018.

[HDWH12] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J Alex Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 205–220, 2012. URL: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/heninger.

[HLCX21] Ching-Hsien Hsia, Shi-Jer Lou, Ho-Hsuan Chang, and Donghua Xuan. Novel hybrid public/private key cryptography based on perfect gaussian integer sequences. *IEEE Access*, 9:145045–145059, 2021. doi:10.1109/ACCESS.2021.3121252.

[HZ10] Yuan He and Wenpeng Zhang. Sum relations for lucas sequences. *Journal of Integer Sequences*, 13(2):3, 2010. URL: https://cs.uwaterloo.ca/journals/JIS/VOL13/He/he7.pdf.

[iac] International association for cryptologic research (iacr). URL: https://www.iacr.org/publications/.

[Ibr09] Bernadin Ibrahimpašić. A cryptanalytic attack on the luc cryptosystem using continued fractions. *Mathematical Communications*, 14(1):103–118, 2009. URL: https://core.ac.uk/download/pdf/14409731.pdf.

[JHW05] Zhengtao Jiang, Yanhua Hao, and Yumin Wang. A new public-key encryption scheme based on lucas sequence. *Journal of Electronics (China)*, 22(5):490–497, 2005. doi:10.1007/bf03037006.

[jis] Journal of integer sequences. URL: https://cs.uwaterloo.ca/journals/JIS/.

[Jon06] G. Jones. Gjrand random numbers, 2006. URL: https://gjrand.sourceforge.net/.

[Jou19] Antoine Joux. Fully homomorphic encryption modulo fermat numbers. 2019. URL: https://eprint.iacr.org/2019/187.

[JSWW76] James P. Jones, Daihachiro Sato, Hideo Wada, and Douglas Wiens. Diophantine representation of the set of prime numbers. *The American Mathematical Monthly*, 83(6):449–464, Jun 1976. doi:10.1080/00029890.1976.11994142.

[JW09] Michael J. Jacobson and Hugh C. Williams. *Solving the Pell equation*. Springer, 2009. doi:10.1007/978-0-387-84923-2.

[Kah13] Reza Kahkeshani. A generalization of the catalan numbers. *Journal of Integer Sequences*, 16(6), 2013. URL: https://cs.uwaterloo.ca/journals/JIS/VOL16/Kahkeshani/kahke3.pdf.

[Kir15] Krishnamurthy Kirthi. Narayana sequences for cryptographic applications. *ArXiv*, 2015. doi:10.48550/arXiv.1509.05745.

[KK16] Krishnamurthy Kirthi and Subhash Kak. The narayana universal code. *arXiv: Information Theory*, 2016. doi:10.48550/arXiv.1601.07110.

[KL21] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2021.

[KLM06] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An introduction to quantum computing*. OUP Oxford, 2006. doi:10.1093/oso/9780198570004.001.0001.

[Knu69] Donald Ervin Knuth. *The art of computer programming. Seminumerical Algorithms.*, volume 2. Addison-Wesley, 1969.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987. URL: https://www.jstor.org/stable/2007884, doi:10.2307/2007884.

[KPVH19] Inas Jawad Kadhim, Prashan Premaratne, Peter James Vial, and Brendan Halloran. Comprehensive survey of image steganography: Techniques, evaluations, and trends in future research. *Neurocomputing*, 335:299–326, 2019. doi:10.1016/j.neucom.2018.06.075.

[Kra86] Evangelos Kranakis. *Primality and cryptography*. Wiley-Teubner Series in Computer Science, 1986. doi:10.1007/978-3-322-96647-6.

[KSC12] D.S. Kumar, C. Suneetha, and A. Chandrasekhar. Novel encryption schemes based on catalan numbers. *International Journal of Engineering Research and Applications (IJERA)*, 2012. URL: https://www.ijera.com/papers/Vol2_issue2/AA22161166.pdf.

[KUH04] Song-Ju Kim, Ken Umeno, and Akio Hasegawa. Corrections of the nist statistical test suite for randomness. *Cryptology ePrint Archive*, 2004. URL: https://eprint.iacr.org/2004/018.pdf.

[LEMV17] Amalia Beatriz Orúe López, Luis Hernández Encinas, Agustín Martín Muñoz, and Fausto Montoya Vitini. A lightweight pseudorandom number generator for securing the internet of things. *IEEE access*, 5:27800–27806, 2017. doi:10.1109/ACCESS.2017.2774105.

[LHA+12] Arjen K Lenstra, James P Hughes, Maxime Augier, Joppe W Bos, Thorsten Kleinjung, and Christophe Wachter. Public keys. In *Annual Cryptology Conference*, pages 626–642. Springer, 2012. doi:10.1007/978-3-642-32009-5_37.

[Lin21] Xin Lin. On the recurrence properties of narayana's cows sequence. *Symmetry*, 13(1):1 – 12, 2021. doi:10.3390/sym13010149.

[LLX+12] Zhihui Li, Bo Lu, Huimin Xu, Gen Li, Wang Li, and Xiang Li. New algorithm for public key cryptosystems based on lucas sequences. In *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE, 2012. doi:10.1109/wicom.2012.6478583.

[LS07] Pierre L'ecuyer and Richard Simard. Testu01: Ac library for empirical testing of random number generators. *ACM Transactions on Mathematical Software (TOMS)*, 33(4):1–40, 2007. doi:10.1145/1268776.1268777.

[LTT95] C.S. Laih, F.K. Tu, and W.C. Tai. On the security of the lucas function. *Information Processing Letters*, 53(5):243–247, 1995. doi:10.1016/0020-0190(94)00209-h.

[LV21] Elena Almaraz Luengo and Luis Javier García Villalba. Recommendations on statistical randomness test batteries for cryptographic purposes. *ACM Comput. Surv.*, 54(4), may 2021. doi:10.1145/3447773.

[Mar95] George Marsaglia. The marsaglia random number cdrom: Including the diehard battery of tests of randomness, 1995. URL: http://ftpmirror.your.org/pub/misc/diehard.

[Mat71] Yuri Matiyasevich. Diophantine representation of the set of prime numbers (in russian). *Dokl. Akad. Nauk SSSR*, (196):770–773, 1971. English translation by R. N. Goss, in Soviet Math. 12 (1971), 249–254. URL: https://www.mathnet.ru/eng/dan35950.

[Mau95] Ueli M. Maurer. Fast generation of prime numbers and secure public-key cryptographic parameters. *Journal of Cryptology*, 8:123–155, 1995. doi:10.1007/bf00202269.

[MC04] Michael Mascagni and Hongmei Chi. Parallel linear congruential generators with sophie–germain moduli. *Parallel Computing*, 30(11):1217–1231, 2004. doi:10.1016/j.parco.2004.08.002.

[McC06] B. D. McCullough. A review of testu01. *Journal of Applied Econometrics*, 21(5):677–682, 2006. URL: http://www.jstor.org/stable/25146455.

[MCPR95] Michael Mascagni, Steven A. Cuccaro, Daniel V. Pryor, and M.L. Robinson. A fast, high quality, and reproducible parallel lagged-fibonacci pseudorandom number generator. *Journal of computational physics*, 119(2):211–219, 1995. doi:10.1006/jcph.1995.1130.

[MD20] Ali Maetouq and Salwani Mohd Daud. Hmnt: hash function based on new mersenne number transform. *IEEE Access*, 8:80395–80407, 2020. doi:10.1109/ACCESS.2020.2989820.

[MHGS21] Souradeep Mukhopadhyay, Sabbir Hossain, Sudipta Kr Ghosal, and Ram Sarkar. Secured image steganography based on catalan transform. *Multimedia Tools and Applications*, 80:14495–14520, 2021. doi:10.1007/s11042-020-10424-4.

[MHK+21] Volodymyr Maksymovych, Oleh Harasymchuk, Mikolaj Karpinski, Mariia Shabatura, Daniel Jancarczyk, and Krzysztof Kajstura. A new approach to the development of additive fibonacci generators based on prime numbers. *Electronics*, 10(23), 2021. doi:10.3390/electronics10232912.

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg. doi:10.1007/3-540-39799-x_31.

[MN98] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998. doi:10.1145/272991.272995.

[MNHS05] Makoto Matsumoto, Takuji Nishimura, Mariko Hagita, and Mutsuo Saito. Cryptographic mersenne twister and fubuki stream/block cipher. *Cryptology ePrint Archive*, 2005. URL: https://eprint.iacr.org/2005/165.

[MS00] Michael Mascagni and Ashok Srinivasan. Algorithm 806: Sprng: A scalable library for pseudorandom number generation. *ACM Transactions on Mathematical Software (TOMS)*, 26(3):436–461, 2000. doi:10.1145/358407.35842.

[MS04] Michael Mascagni and Ashok Srinivasan. Parameterizing parallel multiplicative lagged-fibonacci generators. *Parallel Computing*, 30(7):899–916, 2004. doi:10.1016/j.parco.2004.06.001.

[MS15] Kinga Marton and Alin Suciu. On the interpretation of results from the nist statistical test suite. *Science and Technology*, 18(1):18–32, 2015. URL: http://www.romjist.ro/content/pdf/02-msys.pdf.

[MSH^{+}22] Volodymyr Maksymovych, Mariia Shabatura, Oleh Harasymchuk, Mikolaj Karpinski, Daniel Jancarczyk, and Pawel Sawicki. Development of additive fibonacci generators with improved characteristics for cybersecurity needs. *Applied Sciences*, 12(3):1519, 2022. doi:10.3390/app12031519.

[Mul06] Siguna Muller. Some remarks on williams' public-key crypto functions. *Fibonacci Quarterly*, 44(3):224, 2006. doi:10.1080/00150517.2006.12428314.

[NS02] Nguyen and Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 15:151–176, 2002. doi:10.1007/s00145-002-0021-3.

[oei] The on-line encyclopedia of integer sequences® (oeis®). URL: https://oeis.org/.

[oSC] National Institute of Standards and Technology (NIST). Computer Security Resource Center. Block cipher techniques. URL: https://csrc.nist.gov/Projects/block-cipher-techniques/bcm.

[oSC07] National Institute of Standards and Technology (NIST). Computer Security Resource Center. Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac. 2007. doi:10.6028/NIST.SP.800-38D.

[pri71] *Formulae for the nth prime, Proc. Washington State University Conference on Number Theory 96–107*, 1971.

[pri03] *FIPS publication 180-2: Secure Hash standard.* Number Federal Register (67 FR 54786). National Institute of Standards and Technology (NIST), Jan 2003.

[Pro78] François Proth. Théoremes sur les nombres premiers. *CR Acad. Sci. Paris*, 87:926, 1878.

[PRS12] Fabio Pareschi, Riccardo Rovatti, and Gianluca Setti. On statistical tests for randomness included in the nist sp800-22 test suite and based on the binomial distribution. *IEEE Transactions on Information Forensics and Security*, 7(2):491–505, 2012. doi:10.1109/TIFS.2012.2185227.

[PS21] Rogério Paludo and Leonel Sousa. Number theoretic transform architecture suitable to lattice-based fully-homomorphic encryption. In *2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 163–170. IEEE, 2021. doi:10.1109/ASAP52443.2021.00031.

[Rad72] Charles M. Rader. Discrete convolutions via mersenne transforms. *IEEE Transactions on Computers*, 100(12):1269–1273, 1972. doi:10.1109/T-C.1972.223497.

[RAS+23] Supriadi Rustad, Pulung Nurtantio Andono, Guruh Fajar Shidik, et al. Digital image steganography survey and investigation (goal, assessment, method, development, and dataset). *Signal processing*, 206:108908, 2023. doi:10.1016/j.sigpro.2022.108908.

[Ray20] Prasanta Kumar Ray. A cryptography method based on hyperbolicbalancing and lucas-balancing functions. *Proyecciones (Antofagasta)*, 39(1):135–152, 2020. doi:10.22199/issn.0717-6279-2020-01-0009.

[RBG10] *SP 800-22 Rev. 1 - A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.* National Institute of Standards and Technology (NIST), Apr 2010. doi:10.6028/NIST.SP.800-22r1a.

[RBG15] *SP 800-90A Rev.1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators.* National Institute of Standards and Technology (NIST), Jun 2015. doi:10.6028/NIST.SP.800-90Ar1.

[RBG18] *SP 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation.* National Institute of Standards and Technology (NIST), Jan 2018. doi:10.6028/10.6028/NIST.SP.800-90B.

[RBG23] *NIST IR 8427 - Discussion on the Full Entropy Assumption of the SP 800 90 Series.* National Institute of Standards and Technology (NIST), Apr 2023. doi:10.6028/NIST.IR.8427.

[RBG24] *SP 800-90C - Recommendation for Random Bit Generator (RBG) Constructions (4th Draft).* National Institute of Standards and Technology (NIST), Jul 2024. doi:10.6028/NIST.SP.800-90C.4pd.

[Rib96] Paulo Ribenboim. *The New Book of Prime Number Records, third edition.* Springer-Verlag New York Inc., 1996.

[Row08] Eric S. Rowland. A natural prime-generating recurrence. *Journal of Integer Sequences*, 11(2):28, 2008. URL: https://cs.uwaterloo.ca/journals/JIS/VOL11/Rowland/rowland21.pdf.

[RS01] Ron Rivest and Robert Silverman. Are 'strong' primes needed for rsa. *Cryptology ePrint Archive*, 2001. URL: https://eprint.iacr.org/2001/007.

[RS15] José L Ramírez and Víctor F Sirvent. A note on the k-narayana sequence. *Annales Mathematicae et Informaticae*, 45:91–105, 2015. URL: http://publikacio.uni-eszterhazy.hu/3277/.

[RSA78] Ronald Linn Rivest, Adi Shamir, and Leonard Max Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978. doi:10.1145/359340.359342.

[Rya23] Boris Ryabko. Reduction of the secret key length in the perfect cipher by data compression and randomisation. 2023. URL: https://eprint.iacr.org/2023/1036.

[Saa11] Markku-Juhani O. Saarinen. Sgcm: the sophie germain counter mode. *Cryptology ePrint Archive*, 2011. URL: https://eprint.iacr.org/2011/326.

[SAB18] Muzafer Saračević, Saša Adamović, and Enver Biševac. Application of catalan numbers and the lattice path combinatorial problem in cryptography. *Acta Polytechnica Hungarica*, 2018. doi:10.12700/APH.15.7.2018.7.5.

[SAM$^+$19] Muzafer Saračević, Saša Adamović, Vladislav Miškovic, Nemanja Macek, and Marko Šarac. A novel approach to steganography based on the properties of catalan numbers and dyck words. *Future Generation Computer Systems*, 2019. doi:10.1016/J.FUTURE.2019.05.010.

[SAM$^+$20] Muzafer Saračević, Saša Adamović, Nemanja Macek, Mohamed Elhoseny, and Shahenda Sarhan. Cryptographic keys exchange model for smart city applications. *Iet Intelligent Transport Systems*, 2020. doi:10.1049/IET-ITS.2019.0855.

[SAM$^+$21a] Muzafer Saračević, Saša Adamović, Nemanja Macek, Aybeyan Selimi, and Selver Pepic. Source and channel models for secret-key agreement based on catalan numbers and the lattice path combinatorial approach. *Journal of Information Science and Engineering*, 2021. doi:10.6688/JISE.202103_37(2).0012.

[SAM$^+$21b] Muzafer Saračević, Saša Adamović, V. Miskovic, Mohamed Elhoseny, Nemanja Maček, Mahmoud Mohamed Selim, and Kathiresan Shankar. Data encryption for internet of things applications based on catalan objects and two combinatorial structures. *IEEE Transactions on Reliability*, 2021. doi:10.1109/TR.2020.3010973.

[SC20] Lama Sleem and Raphaël Couturier. Testu01 and practrand: Tools for a randomness evaluation for famous multimedia ciphers. *Multimedia Tools and Applications*, 79(33):24075–24088, 2020. doi:10.1007/s11042-020-09108-w.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949. doi:10.1002/j.1538-7305.1949.tb00928.x.

[She91] O. B. Sheynin. The notion of randomness from Aristotle to Poincaré. *Mathématiques informatique et sciences humaines*, 114:41–55, 1991. URL: http://www.numdam.org/item/MSH_1991__114__41_0/.

[Sho09] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009. doi:10.1017/CBO9780511814549.

[Shp12] Igor Shparlinski. *Number theoretic methods in cryptography: Complexity lower bounds*, volume 17. Birkhäuser, 2012. doi:10.1007/978-3-0348-8664-2.

[Siv20] Ramaswamy Sivaraman. Knowing narayana cows sequence. *Advances in Mathematics: Scientific Journal*, 9(12):10219 – 10224, 2020. doi:10.37418/amsj.9.12.14.

[SK14] Mario Stipčević and Çetin Kaya Koç. *True Random Number Generators*, page 275–315. Springer International Publishing, 2014. doi:10.1007/978-3-319-10683-0_12.

[SKB18] Muzafer Saračević, Edin Korićanin, and Enver Biševac. Encryption based on ballot, stack permutations and balanced parentheses using catalan-keys. *Journal of Information Technology and Applications*, 2018. doi:10.7251/JIT1702069S.

[SL93] Peter J. Smith and Michael J.J. Lennon. Luc: A new public key system. In *Proceedings of the IFIP TC11, Ninth International Conference on Information Security: Computer Security*, IFIP/Sec '93, page 103–117. North-Holland Publishing Co., 1993.

[Slo73] Neil Sloane. *A Handbook of Integer Sequences*. Academic Press, 1973. doi:10.1016/c2013-0-11509-7.

[Smi93] Peter Smith. Luc public key encryption: a secure alternative to rsa. *Dr. Dobb's J.*, 18(1):44–49, jan 1993. URL: https://dl.acm.org/doi/10.5555/156453.156460.

[SOMK22] Marek Sỳs, Lubomír Obrátil, Vashek Matyáš, and Dušan Klinec. A bad day to die hard: Correcting the dieharder battery. *Journal of Cryptology*, 35:1–20, 2022. doi:10.1007/s00145-021-09414-y.

[SP95] Neil Sloane and Simon Plouffe. *The Encyclopedia of Integer Sequences*. Academic Press, 1995.

[SP21] Yanina Y. Shkel and H. Vincent Poor. A compression perspective on secrecy measures. *IEEE Journal on Selected Areas in Information Theory*, 2(1):163–176, 2021. doi:10.1109/JSAIT.2021.3055692.

[Spr04] Renzo Sprugnoli. Combinatorial identities. *Dipartimento di Sistemi e Informatica, Firenze, Italy*, 2004. URL: http://www.dsi.unifi.it/âĹresp/GouldHW.pdf.

[SSA21] Muzafer Saračević, Sudhir Kumar Sharma, and Khaleel Ahmad. A novel block encryption method based on catalan random walks. *Multimedia Tools and Applications*, 2021. doi:10.1007/s11042-021-11497-5.

[SSS18] Muzafer Saračević, Aybeyan Selimi, and Faruk Selimović. Generation of cryptographic keys with algorithm of polygon triangulation and catalan numbers. *Computer Science*, 2018. doi:10.7494/CSCI.2018.19.3.2749.

[SSSK21] Faruk Selimović, Predrag Stanimirović, Muzafer Saračević, and Predrag Krtolica. Application of delaunay triangulation and catalan objects in steganography. *Mathematics*, 9(11):1172, 2021. doi:10.3390/math9111172.

[ST86] John Shawe-Taylor. Generating strong primes. *Electronics Letters*, 22:875–877, 1986. doi:10.1049/el:19860598.

[Sta] Richard P. Stanley. Catalan addendum, preprint, may 25 2013. Available online. URL: http://www-math.mit.edu/~rstan/ec/catadd.pdf.

[Sta99] Richard P. Stanley. *Enumerative Combinatorics, Vol. 2*. Cambridge University Press, 1999. doi:10.1017/CBO9780511609589.

[Sta15] Richard P. Stanley. *Catalan Numbers*. Cambridge University Press, 2015. doi:10.1017/CBO9781139871495.

[Sti05] Douglas R. Stinson. *Cryptography: theory and practice*. Chapman and Hall/CRC, 2005. doi:10.1201/9781420057133-4.

[STM10] L.M. Surhone, M.T. Timpledon, and S.F. Marseken. *Pseudorandom Number Generator*. Betascript Publishing, 2010.

[Tho07] James Harold Thomas. Variations on the fibonacci universal code. 2007. arXiv:cs/0701085, doi:10.48550/arXiv.cs/0701085.

[VZGS13] Joachim Von Zur Gathen and Igor E. Shparlinski. Generating safe primes. *Journal of Mathematical Cryptology*, 7(4):333–365, 2013. doi:10.1515/jmc-2013-5011.

[Wal08] John Walker. Ent—a pseudorandom number sequence test program, 2008. URL: https://www.fourmilab.ch/random/.

[Wil64] C. P. Willans. On formulae for the nth prime number. *The Mathematical Gazette*, 48(366):413–415, 1964. doi:10.2307/3611701.

[WS79] Hugh C. Williams and Bruce K. Schmid. Some remarks concerning the m.i.t. public-key cryptosystem. *BIT Numerical Mathematics*, 19:525–538, 1979. doi:10.1007/bf01931269.

[WS19] Joanne Woodage and Dan Shumow. An analysis of nist sp 800-90a. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part II 38*, pages 151–180. Springer, 2019. doi:10.1007/978-3-030-17656-3_6.

[Yao82] Andrew C. Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.

[Zec72] Édouard Zeckendorf. Representations des nombres naturels par une somme de nombres de fibonacci on de nombres de lucas. *Bulletin de La Society Royale des Sciences de Liege*, pages 179–182, 1972. URL: https://cir.nii.ac.jp/crid/1570009749187075840.

[Zhi13] Lin Zhiqiang. The transformation from the galois nlfsr to the fibonacci configuration. In *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies*, pages 335–339. IEEE, 2013. doi:10.1109/EIDWT.2013.64.

[Çe21] Çağla Çelemoğlu. The third order variant narayana codes and some straight lines corresponding to these. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, 2021. doi:10.29130/DUBITED.1007719.