



HAL
open science

Fast and Private Pool Testing and Contributions to Experimental Mathematics

Ofer Yifrach-Stav

► **To cite this version:**

Ofer Yifrach-Stav. Fast and Private Pool Testing and Contributions to Experimental Mathematics. Computer Science [cs]. École normale supérieure - Paris, 2024. English. NNT: . tel-04513104

HAL Id: tel-04513104

<https://hal.science/tel-04513104v1>

Submitted on 20 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Public Domain

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École normale supérieure

**Fast and Private Pool Testing
and Contributions to Experimental Mathematics**

Soutenu par
Yifrach Stav, Ofer

Le 28 février 2024

École doctorale n°386
**Sciences Mathématiques de
Paris Centre**

Spécialité
Informatique

Préparée à
L'École normale supérieure

Composition du jury :

Peter Y.A. RYAN Professeur, Univ. du Luxembourg	<i>Président</i>
Naila HAYEK Professeure, Univ. Paris-Panthéon-Assas, France	<i>Rapporteuse</i>
Marc JOYE Docteur, HDR, Zama, France	<i>Rapporteur</i>
Jean-Jacques QUISQUATER Professeur, Univ. Catholique de Louvain, Belgique	<i>Examineur</i>
Robert WEIL Directeur de recherche, CNRS CIMIparis, France	<i>Codirecteur</i>
David NACCACHE Professeur, École normale supérieure, France	<i>Codirecteur</i>

Acknowledgements

I would like to begin by expressing my deepest gratitude to my thesis supervisors. Pr. David Naccache and Pr. Robert Weil, without whom this endeavor would not have been possible. David, thank you for your unwavering encouragement, for challenging me when needed, and for being open to play along with my ideas. Robert, thank you for your support and guidance throughout the years.

My sincere thanks are extended to Pr. David Pointcheval for giving me the opportunity to be a member of this department, and to my committee members Dr. Hab. Marc Joye, Pr. Naila Hayek, Pr. Jean-Jacques Quisquater and Pr. Peter Y.A. Ryan for their valuable input and expertise. It is an honor to have such a prestigious committee. I am also very grateful to Academician Pr. Stéphane Mallat, and to Pr. Peter Y.A. Ryan for their valuable feedback and insights throughout my research.

I am grateful to my co-authors Marc Beunardeau, Éric Brier, Noémie Cartier, Aisling Connolly, Nathanaël Courant, Megi Dervishi, Julien Jainsky and Bassem Ouni, with whom I had the pleasure of collaborating. Special thanks are due to Rémi Géraud-Stewart for his hard work, dedication, and continuous support throughout our shared projects. Two exceptional individuals deserve special acknowledgment: Marcel Hollenstein and Peter Rønne, who joined me in the DNA privacy project. Thank you for the enlightening discussions, and for providing valuable comments and advice.

I am also grateful to the colleagues who collaborated with me on the Groupool project, Antoine Deleforge from Inria, Nancy Grand Est, Olivier Gossner from CNRS, École Polytechnique LSE, Bastien Mallein from LAGA, Université Sorbonne Paris Nord, and Jean-François Rupprecht from CNRS, Aix-Marseille University. It was truly remarkable and inspiring to see research professionals from around the world join forces to work together during a global crisis.

Grateful appreciation is due to Pr. Winston Heap for his remarks concerning a preliminary version of “A Note on Moments of Random Multiplicative Functions and Truncated Characteristic Polynomials”. I would also like to thank Dan Brown and Chris Monico on their great insight and pertinent comments during the developments of the ideas listed in the “Fiat–Shamir Goes Tropical” paper. A special note goes to Thibaut Heckmann for his time and effort in proofreading and raising important questions regarding the DNA Privacy section. Aisling Connolly and Georges-Axel Jaloyan also deserve my gratitude for their valuable assistance during challenging periods. In addition, the ongoing support provided by Yuval Silman and Charles Campbell was invaluable to me; I thank you and value your friendship.

Finally, I would like to express my heartfelt gratitude to my family. To my mother, who tirelessly fought for my education: thank you for your love and devotion, and for instilling in me the love for mathematics. To my aunt, for her guidance and unceasing support. To my children Yair and Michael, whose unflinching belief in me has kept my motivation high. And last but not least, to my beloved wife Noga, who has been my rock and

stood by me through every step of this challenging journey. Your enduring love, encouragement, and patience have been my greatest source of strength.

I thank all of you from the bottom of my heart.

Ofer Yitzchak - Stav

“If mathematics describes an objective world just like physics, there is no reason why inductive methods should not be applied in mathematics just the same as in physics.”

-Kurt Gödel

Résumé

Cette thèse est le fruit de recherches menées entre 2019 et 2023, réparties en trois parties. Dans la première partie, nous étudions des questions liées à la pandémie du Covid-19, telles que les tests par lots, un domaine bien établi dans lequel les échantillons de plusieurs patients sont mélangés pour effectuer des tests collectifs. Une telle procédure permet de réduire les coûts et d'économiser du temps. Nous proposons des algorithmes tenant compte des probabilités *a priori* que les tests individuels soient positifs. De telles probabilités peuvent être évaluées lors d'un examen clinique préalable du patient. Nous examinons également les tests par lots en situation d'urgence, où certains échantillons doivent être analysés en priorité. Dans les deux cas, nous proposons de nouveaux algorithmes et les analysons en détail. Cette section traite également de la préservation de la confidentialité de l'ADN dans les tests de dépistage du Covid-19. Dans la deuxième partie, nous présentons nos résultats en mathématiques expérimentales, où nous avons découvert plusieurs nouvelles conjectures sur les fractions continues grâce à des explorations automatisées. Toutes ces conjectures ont été testées numériquement pour évaluer leur plausibilité. Enfin, dans la troisième partie de la thèse, nous abordons divers résultats dans le domaine de la sécurité informatique, tels qu'une attaque inconnue jusqu'à présent sur le logiciel Mathematica, un nouveau mécanisme de protection contre les médicaments contrefaits, et des nouvelles observations sur les preuves à divulgation nulle.

Mots clés : Sécurité de l'information, Tests par lots, Mathématiques expérimentales, Confidentialité de l'ADN, Covid-19

Abstract

This thesis is the culmination of research conducted between 2019 and 2023. It is divided into three parts. In the first part, we explore algorithms related to the Covid-19 pandemic, such as Pool Testing, a well-established technique where samples from multiple patients are pooled for collective testing, allowing for cost reduction and time savings. We propose algorithms taking into account the *a priori* probabilities that individual tests are positive, which can be evaluated during a prior clinical examination of the patient. We also examine Pool Testing in emergency situations, where certain samples need to be analyzed according to some prescribed priority order. In both cases, we propose new algorithms and analyze them in detail. This section also deals with DNA privacy preservation in Covid-19 tests. In the second part, we present our results in experimental mathematics, where we have discovered several new conjectures on continued fractions through automated exploration. All those conjectures have been numerically tested to assess their plausibility. Finally, the third part of this thesis is devoted to various results in the field of computer security, such as a previously unknown attack on the Mathematica software, a new protection mechanism against counterfeit medication, and new observations on zero-knowledge proofs.

Keywords : Information Security, Pool Testing, Experimental Mathematics, DNA Privacy, Covid-19

Contents

Acknowledgements	i
Résumé	v
Abstract	vi
Table of contents	vi
List of Figures	ix
List of Tables	xi
1 Introduction	1
1 Context	1
2 Topical Coverage of this Thesis	1
3 Background and Definitions	7
4 Résumé de la thèse en français	13
2 Covid-19 Related Research	19
1 Introduction and Motivation	19
2 Optimal Covid-19 Pool Testing with <i>A Priori</i> Information	25
3 Near-Optimal Pool Testing Under Urgency Constraints	47
4 Preservation of DNA Privacy During the Large Scale Detection of Covid-19	61
3 Experimental Mathematics	71
1 A Note on the Ramanujan Machine	71
2 Pattern Recognition Experiments on Mathematical Expressions	83
3 On Moments of Random Multiplicative Functions and Truncated Characteristic Polynomials	99
4 A Conjecture From a Failed Cryptanalysis	103
5 The Balkans Continued Fraction	105
4 Practical Contributions to Information Security	163
1 Invisible Formula Attacks	163
2 On Squaring Modulo Mersenne Numbers	171
3 On The Practical Advantage of Committing Challenges in Zero-Knowledge Protocols	173
4 Authenticating Medications with QR-Codes and Compact Digital Signatures	181
5 Fiat-Shamir Goes Tropical	189
5 Conclusions and Perspectives	195

1	Conclusions	195
2	Perspectives and Further Research	197
3	Thesis Defense Reports	199
List of Publications		205
	Peer-Reviewed Journal Articles	205
	Electronic Pre-Prints	205
	Peer-reviewed International Conferences	206
	National Conference and Seminar Presentation	206
	Patents	206
Bibliography		207

List of Figures

1.1	Announcement of new results on the Ramanujan Machine website - January 2023	5
2.1	The “naive procedure” for $n = 2$	25
2.2	Two pooling testing procedures having $(1, 2)$ as root.	26
2.3	Optimality zones for $n = 2$	27
2.4	Graphical representation of a testing procedure.	28
2.5	Slices of the cube decomposition for the $n = 3$ metaprocedure	33
2.6	Slices through the cube at the $z = 0.17$ (left) and the $z = 0.33$ (right) planes	34
2.7	A 3D visualisation of the cube - 52 substructures (looking from $(-1, -1, -1)$).	34
2.8	Naive algorithm, (order of tests unimportant in left and right branches)	36
2.9	Optimal procedures (without permutations) for each zone when $n = 3$	42
2.10	The optimal metaprocedure tree (left), and heuristic metaprocedure (right)	43
2.11	Trees representation with a grouping by one element on the root	43
2.12	Trees representation with a grouping by two elements on the root	44
2.13	Trees representation with a grouping by three elements on the root.	45
2.14	A unary test: draw a random individual from the queue and test it.	48
2.15	A high-level overview of a step during a population test	48
2.16	Algorithm \mathcal{A}_2	49
2.17	Algorithm \mathcal{A}_3	50
2.18	Algorithm \mathcal{A}_4 . Note that D is never re-pooled.	50
2.19	Algorithm \mathcal{A}_5	52
2.20	Region in which each elementary algorithm reaches $> 99\%$ optimality	55
2.21	Algorithm \mathcal{M}_1 . Mixed pair test with re-pooling	57
2.22	Algorithm \mathcal{M}_2 . Mixed pair test with re-pooling	57
2.23	Algorithm \mathcal{M}_3 . Mixed population recursive testing	58
2.24	Testing following DNA destruction	68
3.1	The Five j, k Areas.	142
3.2	The areas of Figure 3.1, with the dimension c added.	143
3.3	Functional dependency between the functions computing $Q_{1,\kappa,c}$	143
3.4	j, κ and c -level master formulae.	143
3.5	κ -level resolution process for Kosovo	144
3.6	Inferring $\bar{\alpha}_{j+2}, \bar{\beta}_{j+2}$ from $\bar{\alpha}_j, \bar{\beta}_j$ and $\bar{\alpha}_{j+4}, \bar{\beta}_{j+4}$	145
3.7	$Q_{j,\kappa,c}$ <i>Requiescat in pace</i>	146
3.8	The axes along which the κ -level master formulae operate in each area.	146
3.9	ϵ, δ for the Balkans and Inostranstvo.	147
3.10	τ, η, μ for the Balkans and Inostranstvo.	147
3.11	A typical Backgammon board.	148

3.12	$Q_{j,\kappa,c}$ <i>Process killers</i>	148
3.13	Steps are ordered from left→right and up→down, i.e. 1 1' 2 33' 4 5 6.	149
3.14	The points on which Montenegro is tested for $1 \leq c \leq 14$	150
3.15	The points on which Bosnia & Herzegovina is tested for $1 \leq c \leq 14$	151
3.16	The points on which the c -level master formula is tested for $1 \leq c \leq 7$	152
3.17	The points on which Kosovo is tested for $1 \leq c \leq 7$	153
3.18	The points on which the Kosovo-Serbia symmetry is tested at the $(\alpha_{j,\kappa}, \beta_{j,\kappa})$ -level.	154
3.19	The points on which Croatia is tested directly at the $(\alpha_{j,\kappa}, \beta_{j,\kappa})$ -level.	155
3.20	The points on which the ratio conjecture is tested for $1 \leq c \leq 7$	156
4.1	The initial notebook.	164
4.2	Executing the notebook.	164
4.3	A False is displayed, as expected	165
4.4	The initial notebook. Flag changed to True to activate the invisible formula.	165
4.5	Executing the notebook again.	166
4.6	n is factored.	166
4.7	A 10-bit random pattern formed naturally during packaging.	182
4.8	Using $k = 2$ stripes to encode information in capsules (left) and Diameter detection (right).	182
4.9	30 pills encoding information using diameter orientation (illustration).	183
4.10	Pill identification attempt in the absence of artefacts.	184
4.11	Pill identification attempt in the presence of artefacts.	185
4.12	Simple paper box with a window.	185
4.13	Micro QR-codes on medications printed on a medication package.	186

List of Tables

2.1	Generation results for some small n	31
2.2	Procedures and metaprocedures for some values of n	33
2.3	Attack and protection scenarii.	65
3.1	Test results	87
3.2	The conjectured relations.	90
3.3	Radicals $\sqrt{\tau}$ appearing for all $i \in \mathbb{N}$ in $Q(0, c_{k,i}, 0)$ for $1 \leq k \leq 15$	91
3.4	Test results	92
3.5	Test results	93
3.6	The first convergence values for $u = 1$	94
3.7	Test results	94
3.8	The first convergence values for $u = 3$	95
3.9	Other convergence values.	96
3.10	Example relations for which $i + j = 2$	97
3.11	Example relations for which $i + j \neq 2$	97
3.12	Test results	97
3.13	Convergence to $\sqrt{6}/\pi$	101
3.14	Areas.	108
3.15	$\psi_1(i, j)$ for $0 \leq i \leq 5$	115
3.16	$\psi_2(i, j)$ for $0 \leq i \leq 5$	116
3.17	Knowledge Summary	117
3.18	Relations for π , G and $\log 2$. See code snippet "15. Series".	121
3.19	The database B	133
3.20	Number of eliminated \vec{u} combinations per c tried.	133
3.21	Examples of convergence to $\frac{a_0}{a_1 + a_2 \log 2}$	137
3.22	Examples of convergence to $\frac{a_0}{a_1 + a_2 \log 2}$	138
3.23	Examples of convergence to $\frac{a_0}{a_1 + a_2 G}$	139
3.24	Examples of convergence to $\frac{a_0}{a_1 + a_2 G}$	140
3.25	Examples of convergence to $\frac{a_0}{a_1 + a_2 \log 2}$ for R_c	140
3.26	The first $\bar{\alpha}_j = \left\{ \left\{ \overset{\alpha}{\alpha}_j, \overset{\beta}{\alpha}_j \right\} \right\}$ values and the first $\bar{\beta}_j = \left\{ \left\{ \overset{\alpha}{\beta}_j, \overset{\beta}{\beta}_j \right\} \right\}$ values.	141
4.1	Protocol modification to accommodate simulation.	191
4.2	Protocol modification to accommodate simulation.	191

Chapter 1

Introduction

1 Context

This PhD thesis presents my scientific results accomplished between 2019 and 2023, which largely coincided with the Covid-19 crisis. I had initially set out to investigate general mathematical and information security problems, but the urgency of the pandemic crisis required an “all hands on deck” approach from the scientific community, and scientists worldwide, myself included, put in efforts to help solve the global crisis.

In this sense, the situation provided an opportunity for me and other ENS PhD candidates to apply our skills to the development of algorithms for optimizing and anonymizing Covid-19 tests. These algorithms not only served a practical purpose but also involved nontrivial and beautiful mathematics, as the reader will soon discover. Despite the challenging nature of the crisis, I was able to make significant contributions in this area, and was able to return to my planned research in other areas as the emergency subsided.

My co-supervisor, David Naccache, assigned me a diverse range of problems from a variety of fields, including number theory, electrics and machine learning. This diversity of work is a vital step in what David Naccache believes a computer scientist’s education needs to include, so he encourages his students to resist sticking to “their” areas of expertise, and to explore others. As a result of this beautiful albeit daunting journey, my research spans across diverse scientific fields and this document is thus composed of three independent parts: Covid-19 related research, including Pool Testing and privacy-preserving tests, results in experimental mathematics and the discovery of new conjectures, and various results in the broad area of computer security.

2 Topical Coverage of this Thesis

2.1 Covid-19-Related Research

The first part of the thesis revolves around the Covid-19 pandemic and its corollary challenges. In March 11, 2020, the World Health Organization (WHO) declared Covid-19 a pandemic [Cucinotta, 2020], making it evident that the world was facing a crisis situation like which we have not seen since the Spanish Flu in 1918 [Robinson, 2021]. As the the number of cases began to rise, the medical world was preparing to support the fast-growing needs of the population, highlighting the need for “out-of-the-box” solutions. Notably, the importance of testing emerged as a key factor in controlling the spread of the virus. Unfortunately, shortages in the Covid-

19 testing supply were consistently reported as a hindrance in the ability to maintain sufficient, consistent, and equitable testing across the population [Beaudevin, 2021].

Rapid Antigen tests were only approved in August 2020, before which testing schemes relied mostly on PCR tests, which required specialized equipment and could not be performed at home, thus adding more load on the already-burdened medical system [Emanuel, 2020]. In addition, many countries out-sourced their testing services, imposing privacy risks, as will be discussed in Chapter 2 (Section 4). Even with the outsourcing, the test shortages continued to be prevalent, and the need to find more efficient and resource-effective test methods remained a priority throughout the pandemic.

2.1.1 Pool Testing

Pool testing (or Group Testing) is a relatively new field of applied mathematics used on various practical applications. The concept of Group Testing is credited to Robert Dorfman (1943) [Dorfman, 1943] who wished to test US servicemen for syphilis. Testing each person individually requires drawing a blood sample from each person and then analysing each sample to determine the presence or absence of syphilis. Since this would have been costly and time-consuming, Dorfman developed a strategy in which several samples could be tested simultaneously. His paper — ‘The Detection of Defective Members of Large Populations’ (1943) is a landmark in the sphere of Combinatorial Group Testing.

In group testing, multiple samples are mixed, and the resulting ‘pool’, is tested using the same amount of material or equipment that would have been required to test one individual sample. In essence, pool testing is concerned with the classification of N units into two categories: “good” and “defective”. Any number n of units ($1 \leq n \leq N$) can be tested simultaneously with only one of two possible outcomes: if all units are “good”, the result will be “good”. However, when at least one sample in the pool is positive, then the pool test receives a “defective” result. This means that (at least) one sample in the pool is “defective”, but the test gives no information about which one it is, or how many “defective items” are in the pool. The most naive approach is then to re-test individually each sample, wasting even more resources.

The units are assumed to have come independently from a binomial population with common probability p of being “defective” and $q = 1 - p$ of being “good”. The challenge in pool testing is to devise a scheme, preferably sequential, using the minimal number of tests needed to classify all of the N units as either “good” or “defective” [Pasternack, 1974].

Throughout the years between Dorfman’s development and the pandemic in 2020, pool testing has already been used to screen large portions of the population in scarcely-infected areas (or as a best-effort measure, when test availability was low). Pool testing has been successfully used to identify viral diseases, such as HIV [Nguyen, 2019], ZIKA [Bierlaire, 2017], and Influenza [Van, 2012]. In addition, pool testing has been suggested as a screening method for routine HCV, HBV, and HIV -1 PCR donors for blood-banks [Roth, 1999].

The idea of using pool testing in light of the global pandemic in 2020 was suggested by many researchers and applied in many countries. By July 2020, it was already in use in dozens of countries as a main testing scheme. But there are various ways of performing pool testing: from halving methods, through triplet testing, matrix testing etc., all in search for the most efficient way to get the maximum information with the minimum of resources.

The section entitled “Optimal Covid-19 Pool Testing with *a priori* Information” describes how to optimally detect infected patients in pools, meaning, using a minimal number of tests to precisely identify them,

given the *a priori* probabilities that each of the patients is healthy. Those probabilities can be estimated using questionnaires, supervised machine learning or clinical examinations. The resulting algorithms, which can be interpreted as informed divide-and-conquer strategies, are non-intuitive and quite surprising.

The section entitled “Near-Optimal Pool Testing under Urgency Constraints” discusses testing strategies that provably approach best-possible strategy - optimal in the sense that no other strategy can give exact results with fewer tests. Our algorithms guarantee that they provide a complete and exact result for every individual, without exceeding $\frac{1}{0.99}$ times the number of tests the optimal strategy would require. This threshold is arbitrary: algorithms closer to the optimal bound can be described, however their complexity increases, making them less practical. Moreover, the way the algorithms process input samples leads to some individuals’ status to be known sooner, thus allowing to take urgency into account when assigning individuals to tests.

2.1.2 DNA Privacy Security

Privacy can be defined as the “claim of an individual to determine what information about himself or herself should be known to others” [Westin, 1967]. Over the last two decades, and specifically with the rise in use of social media, the right to privacy has been a growing concern for many people. In his book *Three Floors Up*, the Israeli author Eshkol Nevo writes “*There are no secrets in the modern era. Everything is bared, aired, shared, Twittered and Flickered; you can Snapchat and WhatsApp and Viber and Wiki. Nothing is secret, privacy is dead, and the funeral will be broadcast on the Reality Channel*” [Nevo, 2017].

While some believe that the idea of privacy is a modern phenomenon and that the protection of privacy has evolved “not despite new technologies, but because of them” [Salecl, 2002], the notion of having a separate *private* sphere is actually very old. The right to privacy is not specifically mentioned in the American Constitution, but a discussion about this concept was documented as early as 1890, in a review article written by Samuel D. Warren and Louis D. Brandeis, in which they argued that the law needed to provide protection against the invasion of an individual’s privacy. This article is considered to have been the beginning of judicial recognition of the right to privacy in American law.

The notion of *medical* privacy, however, predates American law. The Hippocratic oath, which was written around the 5th century, “places an absolute duty on the physician not only to preserve the confidentiality of medical information, but also to observe discretion about general information relating to patients to which they may become privy in social intercourse” [Higgins, 1989].

Alan F. Westin, who is considered to have been “the father of the modern field of privacy law”, described that following a “privacy baseline” in the years 1945-1960, three phases can be distinguished in contemporary privacy development: 1961–1979, 1980–1989, and 1990–2002 [Westin, 2003]. The Privacy Baseline period was characterized by high public trust in government, business, and the non-profit sector and, therefore, general public comfort with the information collection and use activities of those organizations. In these years, privacy limits were generally accepted by the mass media, and the law addressed privacy issues in traditional constitutional and common law concepts. In other words, “Privacy was essentially a third-level social issue—interesting but neither primary nor even secondary in social and political salience” [Westin, 2003].

In the years 1961–1979, with the growth of popular media, the expansion of computers into more and more daily domains, there was a marked rise in physical, psychological, and data surveillance technologies. At the same time, this was an era of increased socio-political turbulence spurred by events such as the war in Vietnam, Watergate, racial discrimination, and other phenomena, which lead to a increased focus on the notion of

individuality, human rights protests and distrust towards governments. Needless to say, this also had an impact on privacy. Westin writes: “In socio-cultural terms, this era saw fundamental change in public and private allocable criteria, limiting overt discrimination based on race, gender, family-life conformity, sexual activity, and the like. This required revising all business and government record systems that embedded the older criteria, transforming these personal characteristics into private matters”.

In 1980-1989 many technological advancements were made, especially in the area of computers, and in particular, the introduction of personal computers (PC). Since PCs were not connected to each other at that time, no significant threat to privacy was posed by the increasing usage of personal computers. The public showed ambivalence toward new information technologies; on one hand, people appreciated the benefits and conveniences of the technologies, while on the other, worried about the potential implications of abuse of these technologies was just beginning to emerge.

The third era of privacy development (1990-2002), according to Westin, is “the period when privacy became a first-level social and political issue” [Westin, 2003]. The development of privacy as an issue became more evident following 9/11. [Norris, 2017]. Other factors which contributed to the debate regarding privacy were the introduction of the internet in the mid 1990s, the development of wireless communication, the development of data-mining software, government programs blocking private use of encryption tools, and the Human Genome Project.

Each person has a unique genome, consisting of a sequence of chromosomes. The Human Genome Project generated the sequence of the human genome, now allowing mapping the sequence of individuals. Genome sequencing reveals a tremendous amount about an individual, more than their medical records [Greenbaum, 2011]. As DNA sequences become understood as information, the privacy of these sequences becomes an area of public concern [Annas, 2004].

During the Covid-19 pandemic, privacy concerns were emerging regarding confinement, tracing and testing. The scientific debate concerning privacy of the Covid-19 tracing efforts was intense, especially focusing on the choice between centralised and decentralised tracing apps. The privacy concerns regarding Covid-19 testing, however, have not received as much attention even though the privacy at stake is arguably even higher. Covid-19 tests require the collection of samples. Those samples possibly contain viral material but inevitably also human DNA. Patient DNA is not necessary for the test but it is technically impossible to avoid collecting it.

The unlawful preservation, or misuse, of such samples at a massive scale may hence disclose patient DNA information with far-reaching privacy consequences. Inspired by the cryptographic concept of *Indistinguishability under Chosen Plaintext Attack*, which will be discussed in further detail later in this introduction, the section entitled “Preservation of DNA privacy during the large scale detection of Covid-19” poses the blueprint of novel types of tests allowing to detect viral presence without leaving persisting traces of the patient’s DNA.

2.2 Experimental Mathematics

Quite early in my work, the interesting online project called The Ramanujan Machine had attracted my attention. I found it interesting because it promised to use the latest advancements in artificial intelligence and machine learning to mimic the thought process of the Indian mathematician Srinivasa Ramanujan, who made significant contributions to number theory and other areas of mathematics in the early 20th century.

The Ramanujan Machine is a specialized software tool that aims to discover new mathematical formulae.

The machine has produced several mathematical conjectures and some of them have been proved to be true, while others remain as unproven. The software was conceptualized and developed by a group of undergraduate students at the Technion under the guidance of a faculty member. The details of the machine were published in [Raayoni, 2021] in 2021.

I found the potential for the project to contribute to the field of artificial intelligence by creating a machine that can think mathematically extremely exciting. We hence looked into the conjectures found by the group and attempted to generalize and explain them. Fortunately, our efforts bore fruit and, in a thread of successive papers included in this manuscript, we explained several of the stated conjectures, found a plethora of new ones that got recognized by the Ramanujan Machine team¹.

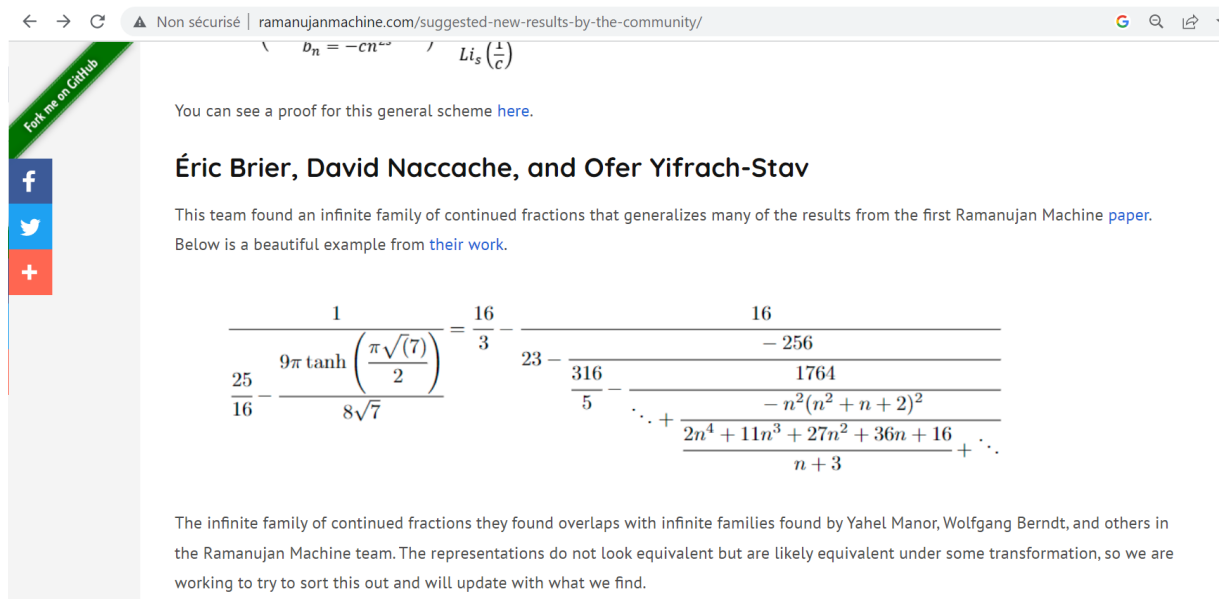


Figure 1.1: Announcement of new results on the Ramanujan Machine website - January 2023

We were pleased to see that the discoveries reported in this thesis were recently reported in a follow-up paper by the Ramanujan Machine [Elimelech, 2023].

2.3 Practical Contributions to Information Security

In this section, the reader will find five contributions to computer security. The first is a conjecture (proved later by [Zhang, 2022]) in algebra. The second is a practical attack on mathematical software presented at Black Hat 2022. We also provide a curious observation about the simulatability of zero knowledge, and an application of short signatures to secure medications against counterfeiting.

¹<http://www.ramanujanmachine.com/suggested-new-results-by-the-community>

3 Background and Definitions

Since this thesis deals with various topics related to combinatorics, information security and experimental mathematics, this section will provide a brief introduction to some of the major notions necessary for the chapters to follow.

3.1 Covid-19-Related Research Background

3.1.1 Indistinguishability under chosen-plaintext attack (IND-CPA)

In order for a cryptosystem to be considered secure, it needs to be able to withhold possible attacks. *The Chosen-Plaintext Attack* is a method of attack that assumes the attacker has access to specific plaintext-ciphertext pairs. This attack aims to extract information that compromises the security of the encryption scheme and uncover details about the encryption key. The notion of *indistinguishability* means that it is impossible for an adversary to distinguish pairs of ciphertexts based on the message they encrypt.

Indistinguishability under chosen-plaintext attack (IND-CPA) means that given an encryption of a message randomly chosen from a two-element message space determined by the adversary, the adversary will not be able to identify the message with a higher probability than a random guess (probability of $\frac{1}{2}$). In other words, should communication between two parties be encrypted, any passive adversary eavesdropping on this communication should be unable to interpret or obtain any information.

Indistinguishability under chosen plaintext attack can be described as a game played between an adversary and a challenger:

- The challenger generates a key-pair pk, sk based on some security parameter k (e.g., a key size in bits), and publishes pk to the adversary. The challenger keeps sk a secret.
- The adversary may perform a polynomially bounded number of encryptions or other operations.
- The adversary hands the challenger two distinct chosen plaintexts: m_0, m_1 .
- After randomly choosing a bit $b \in \{0, 1\}$, the challenger sends back the ciphertext $c = \text{Enc}(pk, m_b)$.
- The adversary may perform any number of additional computations or encryptions.
- The adversary then guesses the value of b .

An attacker is considered to have an *advantage* in distinguishing the ciphertext, if they can successfully distinguish between the chosen ciphertext with a probability significantly greater than $\frac{1}{2}$. Such a scheme is therefore not considered to be secure in terms of indistinguishability.

While this description is specific to an asymmetric key cryptosystem, it can be adapted to the symmetric case.

3.1.2 Information Entropy

Information entropy is a mathematical measure used to assess the level of randomness or uncertainty in a system. It provides a quantification of how much information is needed to describe the outcomes of random events

within the system. When a system exhibits high randomness, the information entropy is also high. On the other hand, in deterministic systems or situations with no randomness, the information entropy is zero.

One of the significant applications of information entropy is in cryptography, where it is used to measure the strength of cryptographic keys or secret information. It allows for the evaluation of the security level of cryptographic algorithms. However, information entropy finds use in various other fields. It is employed in data compression techniques to optimize storage and transmission of information. Additionally, information entropy plays a role in anomaly detection, networking analysis, and other security-related contexts.

The concept of information entropy is closely related to the occurrence distribution of events in a system. It depends on the number of possible outcomes (alphabet size) and the probabilities associated with each outcome.

3.2 Experimental Mathematics

Experimental mathematics is the practice of using mathematical computations to investigate and explore mathematical objects, rather than just using them to prove mathematical notions. This approach utilizes the power of computers to run complex calculations and simulations, often using a trial-and-error method, to discover patterns, identify numbers and sequences, and gather evidence to support specific mathematical assertions.

One of the main goals of experimental mathematics is to use computation to gain insight into mathematical phenomena that might be difficult or impossible to understand using traditional mathematical methods. For example, using computer simulations to explore the properties of large or complex mathematical objects, testing the validity of mathematical conjectures using numerical calculations, or using computational algebraic techniques to prove theorems.

Another important aspect of experimental mathematics is the use of visualization techniques to help understand mathematical concepts and problems. For example, creating graphical representations of mathematical objects or data, or using animation and other interactive tools to explore mathematical phenomena. This field has become increasingly important in recent years, as it allows mathematicians to study problems that are too difficult to solve using traditional methods.

3.2.1 The Birthday Paradox

The Birthday Paradox, also known as *The Birthday Problem* is a veridical paradox coming from probability theory, in which the actual probability is counter-intuitive, and therefore surprising when calculated. The question is quite simple: given a set of n randomly chosen people, what is the probability that at least two will share a birthday? The probability of a shared birthday exceeds 50% in a group of only 23 people. This result is achieved by comparing birthday dates between every possible pair of individuals. With 23 individuals, there are $\frac{23 \times 22}{2} = 253$ pairs to consider. Given that there are 365 days a year, this number is greater than half the possible birthdays.

The birthday problem is commonly attributed to Harold Davenport, who came across it around 1927. However, Davenport did not publish his findings at the time, and did not take credit for discovering it as he believed it must have been previously stated. The first published account of the problem appeared in 1939, by Richard von Mises [Von Mises, 1939].

Since then, the birthday problem has been studied by many mathematicians and statisticians, and has been applied in various fields, including probability theory, statistics, and computer science. The problem has also

been generalized to include various variations, such as the birthday problem with different calendars, or with different numbers of days in a year.

The Birthday paradox has implications relevant to a cryptographic attack called the *Birthday Attack*. The *Birthday Attack* uses a probabilistic model that reduces the complexity of finding a collision (like a shared birthday) for a hash function, and calculating risks of hash collisions in a given size of population. For example, see [Wagner, 2002].

3.2.2 Mersenne Numbers

A Mersenne number is a positive integer that is one less than a power of two. These numbers are named after the French mathematician Marin Mersenne, who studied them in the 17th century. An example of a Mersenne number is $2^3 - 1 = 7$, but there are infinitely many Mersenne numbers. Mersenne numbers have a number of interesting properties, and have been studied in many branches of mathematics, including number theory, cryptography, and computer science. Not all values of the exponent (p) yield a prime Mersenne number; A Mersenne number $M_p = 2^p - 1$ is prime if and only if p is prime.

3.2.3 Continued Fractions

Continued fractions are a remarkable and unique representation of real numbers. They are expressed as an infinite or finite sequence of fractions, with an integer as the first term and subsequent terms as fractions with integer denominators and often unit numerators. Unlike decimal or fractional representations, continued fractions use an infinite sequence of nested fractions to approximate a real number. For instance, the continued fraction representation of a real number x is denoted as $[a_0; a_1, a_2, a_3, \dots]$, where a_0 is the integer part, and a_1, a_2, a_3, \dots represent the successive terms. An example of a continued fraction is $\sqrt{2} = [1; 2, 2, 2, \dots]$. The history of continued fractions dates back to the ancient Greeks, with the great mathematician Euclid being credited with the first known algorithm for computing continued fractions. Later, the Indian mathematician Bhaskara II made significant contributions to continued fractions in the 12th century, followed by the work of mathematicians like Leonhard Euler and Joseph Lagrange in the 18th century.²

One of the most remarkable properties of continued fractions is their ability to provide the best rational approximations for real numbers. Each convergent of a continued fraction $[a_0; a_1, a_2, \dots, a_n]$ is an increasingly accurate rational approximation of x . In fact, the convergents of a continued fraction are the best possible rational approximations, meaning that no other fraction with a smaller denominator can be closer to the real number x .

Example: The continued fraction representation of the irrational number $\sqrt{2}$ is $[1; 2, 2, 2, \dots]$, which can

²Note that Euler was the PhD supervisor of Lagrange, who was the supervisor of Poisson, who was the supervisor of Chasles, who was the supervisor of Darboux, who was the supervisor of Picard, who was the supervisor of Hadamard, who was the supervisor of Fréchet, who was the supervisor of Fortet, who was the supervisor of Cohen, who was the supervisor of David Naccache, who is my co-supervisor.

be written as:

$$\sqrt{2} = 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \frac{1}{\ddots}}}}$$

Continued fractions possess fascinating properties closely tied to the Euclidean algorithm, which applies to both integers and real numbers. Any rational number p/q can be represented by two related expressions as a finite continued fraction, wherein the coefficients a_i are derived through the application of the Euclidean algorithm to the pair (p, q) . The numerical value of an infinite continued fraction is irrational, defined as the limit of a sequence of finite continued fraction values extracted from its infinite sequence of integers. Each finite continued fraction in the sequence is constructed by utilizing a finite prefix of the infinite continued fraction's defining integer sequence. Furthermore, every irrational number α can be expressed as a unique infinite regular continued fraction, with coefficients obtained through the non-terminating version of the Euclidean algorithm applied to the incommensurable values α and 1. This method of representing real numbers, whether rational or irrational, is known as their continued fraction representation.

Continued fractions play a crucial role in various mathematical algorithms. They are employed in the fields of cryptography, signal processing, and error correction codes, among others. The theory of continued fractions is also deeply connected to the Farey fractions, which are rational numbers with denominators bounded by a fixed positive integer.

3.2.4 Catalan Constant and Catalan Numbers

The Catalan constant, denoted as G , K , or C , is a significant mathematical constant found in various mathematical fields, including number theory and combinatorics. It is named after the Belgian mathematician Eugène Charles Catalan.

G is approximately equal to 0.91596559. It appears in diverse mathematical contexts, but one notable area is its application in counting specific combinatorial structures. For instance, it helps calculate the number of valid arrangements of parentheses in well-formed expressions. Beyond its role in combinatorics, the G is related to other mathematical constants and functions. It arises in the evaluation of definite integrals and in the context of polylogarithm functions. While G may not enjoy the same widespread recognition as other mathematical constants, it holds significant mathematical meaning and finds applications in various mathematical disciplines, making it an intriguing constant.

On the other hand, Catalan numbers form a sequence of natural numbers that have various applications in combinatorial problems. These numbers, first studied by Catalan in the 19th century, follow a specific recursive formula and represent the count of valid combinations or structures in different scenarios. For example, Catalan numbers come into play in parentheses expressions, where they represent the number of properly nested arrangements of n pairs of parentheses. Additionally, they are associated with Dyck paths [Labelle, 1990], which are paths consisting of n up steps and n down steps that never go below the x -axis. Triangulations of convex polygons are also linked to Catalan numbers. The sequence of Catalan numbers starts with 1, 1, 2, 5, 14, 42, 132, and continues infinitely. Each Catalan number depends on the previous numbers, resulting in intriguing pat-

terns and relationships. These numbers find applications in various fields, including computer science, discrete mathematics, and combinatorial optimization. Their remarkable properties make them a fascinating subject of study in their own right, continuing to intrigue mathematicians worldwide.

In that respect, I would like to quote an excerpt of a very recent beautiful recent paper [Sloane, 2023] by Sloane:

“I could have chosen a simpler example, like the Fibonacci numbers, but I have a particular reason for choosing the Catalan numbers. When the OEIS was new, people would sometimes say to me that they had a sequence they were trying to understand, and would I show them how to use the database. At least twice when I used the Catalan sequence as an illustration, they said, ‘Why, that is my sequence! How on earth did you know?’ It was no mind-reading trick. The Catalan numbers are certainly the most common sequence that people don’t know about. This entry is the longest – and one of the most important – in the whole database.”

4 Résumé de la thèse en français

4.1 Recherches liées au Covid-19

Alors que l'humanité peine à contenir l'infection mondiale de la Covid-19, les actions prophylactiques sont grandement ralenties par la pénurie de kits de dépistage. Les gouvernements ont pris plusieurs mesures pour pallier cette pénurie : la FDA est devenue plus libérale dans l'approbation des tests de la Covid-19 aux États-Unis. Au Royaume-Uni, des mesures d'urgence ont permis d'augmenter le nombre quotidien de kits de test produits localement à 100 000. La Chine a récemment lancé un vaste programme de fabrication de tests. Cependant, tous ces efforts sont très insuffisants et de nombreux pays pauvres sont encore menacés. Une méthode populaire pour réduire le nombre de tests consiste à regrouper des échantillons, c'est-à-dire à mélanger des échantillons de patients et à tester les échantillons mélangés une fois. Si tous les échantillons sont négatifs, le regroupement réussit à un coût unitaire. Cependant, si un seul échantillon est positif, l'échec n'indique pas quel patient est infecté.

Le premier article de ce chapitre décrit comment détecter de manière optimale les patients infectés dans des groupes, c'est-à-dire en utilisant un nombre minimal de tests pour les identifier précisément, compte tenu des probabilités a priori que chacun des patients soit en bonne santé. Ces probabilités peuvent être estimées à l'aide de questionnaires, d'apprentissage automatique supervisé ou d'examen cliniques. Les algorithmes résultants, qui peuvent être interprétés comme des stratégies de diviser pour mieux régner, sont non intuitifs et assez surprenants.

Le second article constituant ce chapitre adresse des stratégies de test qui approchent de manière prouvée la meilleure stratégie possible - optimale dans le sens où aucune autre stratégie ne peut fournir des résultats exacts avec moins de tests. Nos algorithmes garantissent un résultat complet et exact pour chaque individu, sans dépasser fois le nombre de tests que la stratégie optimale nécessiterait. Ce seuil est arbitraire : des algorithmes plus proches de la limite optimale peuvent être décrits, mais leur complexité augmente, ce qui les rend moins pratiques. De plus, la manière dont les algorithmes traitent les échantillons permet de connaître plus tôt le statut de certains patients, ce qui permet de tenir compte de l'urgence lors de l'attribution des patients aux tests.

Enfin, le chapitre adresse les préoccupations concernant la vie privée émergent du confinement, du traçage et des tests. Le débat scientifique concernant la vie privée des efforts de traçage de la Covid-19 fut intense, en se concentrant notamment sur le choix entre les applications de traçage centralisées et décentralisées. Les préoccupations concernant la confidentialité de l'ADN du patient, cependant, n'ont cependant pas reçu autant d'attention, même si la menace potentielle vie privée en jeu est sans doute encore plus élevée. Les tests de la Covid-19 nécessitent la collecte d'échantillons. Ces échantillons contiennent éventuellement du matériel viral mais inévitablement aussi de l'ADN humain. L'ADN du patient n'est pas nécessaire pour le test, mais il est techniquement impossible d'éviter de le collecter. La conservation illégale, ou l'utilisation abusive, de ces échantillons à grande échelle peut donc divulguer des informations sur l'ADN du patient avec des conséquences de grande portée. Inspiré par le concept cryptographique « d'indistinguabilité sous attaque à clair choisi », cet article pose les bases de nouveaux types de tests permettant de détecter la présence virale sans laisser de traces persistantes de l'ADN du patient.

4.2 Mathématiques expérimentales

Le projet de la Machine Ramanujan détecte de nouvelles expressions liées à des constantes d'intérêt, telles que les valeurs de la fonction ζ , γ et des nombres algébriques (pour n'en citer que quelques-uns).

En particulier, le projet énumère un certain nombre de conjectures impliquant des valeurs de fonction ζ paires et impaires, des logarithmes, etc.

Nous montrons que de nombreuses relations détectées par le projet de la Machine Ramanujan découlent d'une observation algébrique spécifique et montrons comment en générer un nombre infini. Ainsi, un premier article dans ce chapitre fournit une preuve automatisée et/ou une explication de nombreuses relations répertoriées comme conjectures par le projet (bien que pas toutes). Le second article constituant ce chapitre présente les résultats d'expériences de reconnaissance de motifs sur des expressions mathématiques. Nous donnons quelques exemples de résultats conjecturés. Aucun d'entre eux n'a été vérifié en détail pour son caractère novateur. Nous n'avons pas tenté de prouver toutes les relations trouvées et nous nous sommes concentrés sur leur génération.

Une troisième contribution concerne un article [Heap, 2015] où Heap et Lindqvist donnent une formule asymptotique pour le $2k$ -ème moment d'une somme de variables de Steinhaus multiplicatives. Cela a fourni une équivalence asymptotique avec les moments des variables de Steinhaus. Dans [Heap, 2015], une conjecture est formulée sur une espérance liée à un ensemble de variables aléatoires X_p , réparties uniformément sur le cercle de l'unité avec une variance de 1 pour les nombres premiers p . Cette note améliore la borne donnée dans [Heap, 2015].

On notera qu'il est maintenant connu [Harper, 2020] que $\mathbb{E}|S_x| \asymp \frac{\sqrt{x}}{\log \log x}$.

La quatrième contribution du chapitre décrit une observation découverte lors d'une tentative infructueuse de cryptanalyse.

Soit $P(x, y)$ un polynôme bivarié avec des coefficients dans \mathbb{C} . Formons les matrices $n \times n$ L_n dont les éléments sont définis par $P(i, j)$. Définissons les matrices $M_n = L_n - \text{ID}_n$.

Il semble que $\mu(n) = (-1)^n \det(M_n)$ soit un polynôme en n que nous n'avons pas caractérisé.

Nous fournissons un exemple numérique.

La cinquième contribution du chapitre concerne les fractions continues ayant pour numérateur $(v + 1 + n)(v + 2 + n)(v + 3 + n)(v + 4 + n)$.

Soit $a_{n,v} = (v + 1 + n)(v + 2 + n)(v + 3 + n)(v + 4 + n)$ pour $v \in \{0, -1\}$ et soit $b_{n,v,c,t} = c((3 + v + n)^2 - t)$ pour $t \in \{0, 1\}$.

Nous considérons les fractions continues :

$$Q(v, c, t) = \mathcal{K}_{n=1}^{\infty} \left(\frac{a_{n,v}}{b_{n,v,c,t}} \right)$$

En notant $d = \sqrt{c^2 + 4}$, nous observons que :

$$Q(v, c, t) = \frac{t}{2}(d - c)(5v + 8) + \frac{(t - 1)(v + 3)(-3c^3 + 3\sqrt{d^3} + c(v - 13))}{c^2(v - 2) - (v - 3)^2}$$

Comme nous ne fournissons pas une preuve des relations données ici, nous ne prétendons pas qu'elles sont des théorèmes, cependant elles ont été intensivement vérifiées par machine

Enfin, ce chapitre de la thèse se termine par l'étude détaillée d'une collection de fractions continues impliquant la constante de Catalan. Ce dernier article fournit des formules générales gouvernant ces fractions continues. En distinguant différents cas associés à des régions dans le plan, nous surnomons ces fractions continues « Les Balkans », car elles se divisent en zones qui sont liées mais néanmoins différentes par nature.

Là aussi, étant donné que nous ne fournissons pas de preuves formelles de ces formules construites par machine, nous ne prétendons pas qu'elles soient des théorèmes. Néanmoins, chaque formule proposée a été testée de manière extensive numériquement.

Soit $G = 0.91596559\dots$ la constante de Catalan et soit C_n le n -ème nombre de Catalan.

Notons, pour n impair:

$$n!! = \prod_{k=1}^{\frac{n+1}{2}} (2k-1)$$

Définissons pour j impair et $\kappa, c \in \mathbb{N}$ la fraction continue:

$$Q_{j,\kappa,c} = j(2-j+2\kappa) + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n(c+n)(j+n-1)(1-j+2\kappa+n)}{j(2-j+2\kappa) + (3+4\kappa)n + 3n^2} \right)$$

Le résultat, auquel il est fait référence, est un procédé permettant de calculer $Q_{j,\kappa,c}$ de manière exacte. Ce procédé, dont la mise au point fut très complexe, est le suivant:

Si $j \geq 2\kappa + 3$: calculer $Q_{j,\kappa,c}$ par sommation directe:

$$Q_{j,\kappa,c} = j(2-j+2\kappa) + \mathcal{K}_{n=1}^{j-2\kappa-1} \left(\frac{-2n(c+n)(j+n-1)(1-j+2\kappa+n)}{j(2-j+2\kappa) + (3+4\kappa)n + 3n^2} \right)$$

Si $3 + \kappa \leq j \leq 2\kappa + 1$:

Effectuer le changement de variable $j' = 2(\kappa + 1) - j$ et calculer $Q_{j',\kappa,c}$ à l'aide des procédés suivants:

Si $j = 1$: Définir:

$$\Delta_{\kappa,c}(\alpha, \beta) = \begin{cases} \alpha + \beta c & \text{si } c < 2 \\ -2c(2c-1)(2(c-\kappa)-1)^2 \Delta_{\kappa,c-2}(\alpha, \beta) & \text{si } c \geq 2 \\ + (8c^2 + (2-8\kappa)c - 2\kappa + 1) \Delta_{\kappa,c-1}(\alpha, \beta) & \end{cases}$$

$$\Gamma_{\kappa,c}(\alpha, \beta) = (2c-1)!!^2 G + \Delta_{\kappa,c-1}(\alpha, \beta) \cdot \prod_{i=0}^{\kappa-1} (2(c-i)-1)$$

$$\delta_{\kappa} = \frac{4^{\kappa-1}}{(2\kappa-1)C_{\kappa-1}} \text{ et } \rho_{\kappa} = \frac{\delta_{\kappa}(-1)^{\kappa}(1-2\kappa)}{(2\kappa)!(2\kappa-3)!!}$$

$$\alpha_{\kappa} = \rho_{\kappa} \Delta_{1,\kappa-1}(1, -2) \text{ et } \beta_{\kappa} = -\rho_{\kappa} (2\kappa-3)^2 \Delta_{2,\kappa-1}(1, 12) - \alpha_{\kappa}$$

$$\text{Retourner } Q_{1,\kappa,c} = \frac{\delta_{\kappa}(2c)!}{\Gamma_{\kappa,c}(\alpha_{\kappa}, \beta_{\kappa})}$$

Si $3 \leq j \leq \kappa + 2$:

Définir:

$$\varrho(j) = 2^j \left(\frac{j-1}{2}! \right)^2 C_{\frac{j-3}{2}}$$

$$\alpha_{j+2} = \begin{cases} -1 & \text{si } j = 3 \\ 4 \frac{\left(\alpha_j (j-4)(j-1)j(2j-5)(2j-3) - (3j-2)\varrho(j) \right)}{(j-4)(j-1)(j+1)} & \text{si } j > 3 \end{cases}$$

$$\beta_{j+2} = \begin{cases} 115j - 341 & \text{si } 3 \leq j \leq 5 \\ 4 \frac{\left(\beta_j j(2j+1)(j-1)(2j-5)(j-4) - (3j^3 - 17j^2 + 11j + 18)\varrho(j) \right)}{(j-4)(j-3)(j+1)} & \text{si } j > 5 \end{cases}$$

$$\alpha_{\beta_{j+2}} = \begin{cases} -\frac{1}{3} & \text{si } j = 3 \\ 4 \frac{\left(\beta_j (j-6)(j-4)(j-2)j(j-1)(2j-7)(2j-5) - 12(j^2 - 4j + 2)\varrho(j) \right)}{(j-6)(j-4)^2(j-1)(j+1)} & \text{si } j > 3 \end{cases}$$

Définir:

$$\vartheta_1(j) = (j-6)(j-4)(j-2)(j-1)j(2j-7)(2j-5)(-1-3j+2j^2)$$

$$\vartheta_2(j) = 4(-390 + 312j + 653j^2 - 942j^3 + 442j^4 - 87j^5 + 6j^6)$$

$$\vartheta_3(j) = (j-6)(j-4)^2(j-1)(1+j)(13-11j+2j^2)$$

$$\beta_{\beta_{j+2}} = \begin{cases} -\frac{14}{3} & \text{si } j = 3 \\ 4 \frac{\left(\beta_j \vartheta_1(j) - \vartheta_2(j)\varrho(j) \right)}{\vartheta_3(j)} & \text{si } j > 3 \end{cases}$$

A l'aide de ces formules, itérer sur j pour calculer $\alpha_j, \alpha_{\beta_j}, \beta_j, \beta_{\beta_j}$.

Définir:

$$\pi(j, \kappa) = \prod_{i=0}^{\frac{j-3}{2}} (\kappa - i)(2\kappa - 2i - 1)^2$$

$$\ell(n, j, \kappa) = \frac{(-1)^{\kappa+1}(2\kappa)!^2}{\kappa! 2^{3\kappa-2} (2\kappa-j)(2\kappa-1)(n((2\kappa-j-2)(3-2\kappa)-1)+1) \cdot \pi(j, \kappa)}$$

$$\eta(n, j, \kappa) = (2\kappa + 2j - 9 - 2n)(2\kappa + j - 8 - 2n)(-2\kappa + 5 - j)(2\kappa + j - 6)$$

$$\phi(n, j, \kappa) = 8\kappa^2 + \kappa(10j - 48 - 8n) + 3j^2 - (28 + 4n)j + 68 + 18n$$

$$\bar{\Delta}_{n,j,\kappa}(\alpha, \beta) = \begin{cases} \alpha + \beta & \text{si } \kappa < 2 \\ \eta(n, j, \kappa) \cdot \bar{\Delta}_{n,j,\kappa-2}(\alpha, \beta) + \phi(n, j, \kappa) \cdot \bar{\Delta}_{n,j,\kappa-1}(\alpha, \beta) & \text{si } \kappa \geq 2 \end{cases}$$

À l'aide des valeurs $\alpha_j, \alpha_j, \beta_j, \beta_j$ calculer:

$$\alpha_{j,\kappa} = \frac{\bar{\Delta}_{0,j,\kappa-j+2}(\alpha_j, \alpha_j)}{\ell(0, j, \kappa)} \quad \text{et} \quad \beta_{j,\kappa} = \frac{\bar{\Delta}_{1,j,\kappa-j+2}(\beta_j, \beta_j)}{\ell(1, j, \kappa)} - \alpha_{j,\kappa}$$

Définir:

$$\Delta_{j,\kappa,c} = \begin{cases} \alpha_{j,\kappa} + \beta_{j,\kappa}c & \text{si } c < 2 \\ -2c(2c-j)(2c-2\kappa+j-2)(2c-2\kappa-1)\Delta_{j,\kappa,c-2} & \text{si } c \geq 2 \\ + (8c^2 + (2-8\kappa)c + (j-2)(2\kappa-j))\Delta_{j,\kappa,c-1} & \end{cases}$$

$$f_{j,\kappa,c} = C_{\frac{j-3}{2}} C_{\kappa-1} (j-2)(2\kappa-1)(2c-1)!!^2 \prod_{i=1}^{\frac{j-1}{2}} (2c-2\kappa+2i-1)(\kappa-i+1)$$

$$g_{j,\kappa,c} = (2c)! 2^{\frac{j+4\kappa-7}{2}} \prod_{i=1}^{\frac{j-1}{2}} (2c-2i+1)(2\kappa-2i+1)$$

$$h_{j,\kappa,c} = \prod_{i=0}^{\frac{j-3}{2}} (2c-2i-1) \prod_{i=0}^{\kappa-1} (2c-2i-1)$$

Retourner:

$$Q_{j,\kappa,c} = \frac{g(j, \kappa, c)}{\Delta_{j,\kappa,c-1} \cdot h(j, \kappa, c) + f(j, \kappa, c) \cdot G}$$

4.3 Contributions pratiques à la sécurité informatique

La première contribution du chapitre introduit un nouveau vecteur d'attaque applicable à un outil de calcul symbolique couramment utilisé par les cryptographes (Mathematica).

L'attaque tire parti du fait que l'interface utilisateur très riche de cet outil permet d'afficher des formules en couleur invisible ou en taille de police zéro. Cela permet de rendre invisibles certaines parties du code lorsqu'elles sont ouvertes à l'aide de l'outil.

Nous mettons en œuvre une attaque par fautes classique grâce à ce mécanisme trompeur, mais d'autres attaques cryptographiques ou non cryptographiques (par exemple, le formatage du disque de la victime ou l'installation de rootkits) peuvent être facilement menées en utilisant des techniques identiques.

Cela souligne l'importance de créer des logiciels de détection de logiciels malveillants pour les outils de calcul symbolique. De telles protections n'existent pas à ce jour.

Nous insistons sur le fait que notre observation n'est pas une vulnérabilité dans Mathematica mais plutôt un abus des riches possibilités offertes par le logiciel.

La seconde contribution du chapitre fut observée pendant la conception d’une nouvelle primitive inspirée par Squash. Ce faisant, nous sommes accidentellement tombés sur l’observation décrite dans cette partie de la thèse.

Soit n un nombre de Mersenne de k bits dont les facteurs sont inconnus. Considérons un nombre secret x de ℓ bits tel que $x = 2^{k/2}a + b$. Nous observons qu’il existe des configurations de paramètres où une partie de la valeur b^2 est divulguée même si $k < 2\ell$.

Cette observation ne met en danger aucun schéma connu et en particulier pas Squash.

La troisième contribution du chapitre concerne la transformation de Fiat-Shamir. La transformation de Fiat-Shamir est une technique classique permettant de transformer tout Σ -protocole à connaissance nulle en un schéma de signature.

Essentiellement, l’idée sous-jacente à cette transformation est le fait que dériver le défi du hachage de l’engagement supprime la possibilité de simulation et fournit donc des preuves non interactives d’interaction.

Il découle de cette observation que si l’on souhaite préserver la réfutabilité, la taille du défi (par tour) doit être maintenue basse. Par exemple, dans le protocole Fiat-Shamir original, les auteurs recommandent 18 bits mais suggèrent que la taille du défi puisse être augmentée pour réduire l’effort de communication, par exemple la valeur de 20 est proposée dans [Micali, 1990].

Nous montrons qu’avec des tailles de défi relativement petites, la réfutabilité pratique peut être détruite en contraignant artificiellement le vérifieur à utiliser une fonction de hachage ralentie ou en recourant à une agence de confiance proposant un service de destruction la réfutabilité.

La quatrième contribution du chapitre décrit une méthode pour protéger les médicaments contre la falsification, un problème ayant un fort impact en matière de santé publique.

Nous combinons plusieurs technologies existantes pour atteindre cet objectif. Les éléments de base utilisés sont le hasard physique inhérent généré pendant le processus d’emballage, la vision artificielle, de courtes signatures numériques et des codes QR.

Enfin, la thèse se termine par une observation sur un récent ePrint où Brown et Monico [Brown, 2023] proposent de nouvelles attaques contre le schéma de signature tropical de Chen, Grigoriev et Shpilrain [Chen, 2023]. Nous proposons une nouvelle contre-mesure contre ces attaques. Ainsi, nous déplaçons (temporairement ?) le feu de l’algorithme de signature pour rediriger les attaques sur le problème de la clé et sur la factorisation des polynômes tropicaux.

Chapter 2

Covid-19 Related Research

1 Introduction and Motivation

1.1 Testing for Covid-19 Infection

In 2020, the Covid-19 (Coronavirus Disease 2019) pandemic was rapidly spreading, significantly impacting healthcare systems. As the readers know, stay-at-home and social distancing orders were enforced in many countries in an attempt to the control of the disease's spread, while causing turmoil in the economic balance and in social structures [Baker, 2020]. Rapid detection of cases and contacts was an essential component in controlling the spread of the pandemic. In the U.S., the estimations were that at least 500,000 Covid-19 tests were needed to be performed **daily** in order to successfully reopen the economy [Lee, 2020]. Unfortunately, as humanity attempted to limit the global Covid-19 infection, prophylactic actions were grandly slowed-down by the severe shortages of Covid-19 testing kits [Ellis, 2020].

In order to put things in context, let us first provide a short background on tests for Covid-19. There are two main types of tests for Covid-19 [FDA, 2023]:

- *Diagnostic tests* that detect the presence of SARS-COV-2 nucleic acids in human samples. A positive result of these tests indicates the presence of the virus in the body. Covid-19 diagnostic tests are divided into two main sub-types:
 - Molecular tests, such as polymerase chain reaction (PCR) and other nucleic acid amplification tests (NAATs) tests, which detect genetic material called RNA from the virus.
 - Antigen tests (rapid tests), which detect proteins called antigens from the virus.
- *Antibody (Serological) tests* that identify antibodies (e.g., IgM, IgG) to SARS-COV-2 in clinical specimens [Wang, 2020]. Serological tests can be helpful in identifying not only those who are ill, but also those who have been infected, as antibodies are still present in their blood. This identification may be important for several reasons. First, this test can differentiate those who are immune to the virus and those who are still at risk. Secondly, identifying populations who have antibodies can facilitate research on the use of convalescent plasma in the development of a cure for Covid-19 [Hahn, 2020].

As mentioned, at the early stages of the pandemic, both tests were in very short supply. Governments were taking several measures to work around this shortage: the FDA¹ had become more liberal on the approval of Covid-19 tests via the EUA (Emergency Use Authorization) [Hahn, 2020]; in the U.S. and the U.K. there were attempts to boost the number of locally produced test kits to reach a throughput of 100,000 kits per day. Those efforts could not, however, be followed by many countries and entire swaths of Africa, Asia and Latin America appeared to be under concrete threat.

1.2 Pool Testing

One of the strategies which were considered in order to optimize the use of available tests, reduce costs and save time, was *Pool Testing* (also called *Group Testing*). The concept is credited to Dorfman [Dorfman, 1943] who suggested it to detect syphilis in the U.S. military. In the technique of pool testing, several samples are combined and analyzed together using the same resources and materials that would have been needed to test each sample individually. This approach allows for efficient testing of multiple samples simultaneously while conserving resources. However, when at least one sample in the pool is positive, then the pool test fails. This means that (at least) one sample in the pool is positive, but the test gives no information about which one. The most naive approach is then to re-test individually each sample, which can be costly and time consuming.

In Dorfman's approach, pools of identical sizes are formed, and positive pools are retested one by one. Using pools of size n , for a homogeneous population of N individuals and a positive probability p , Dorfman's method performs on average

$$\frac{N}{n}(1 + n(1 - (1 - p)^n))$$

tests. This can be inverted to yield the optimal pool size n^* , which maximizes the number of tested individuals. Dorfman shows that

$$n^* = \frac{2}{\ln(1 - p)} W\left(-\frac{1}{2}\sqrt{-\ln(1 - p)}\right)$$

where W is the Lambert W function².

Following Dorfman, many variants and improvements were suggested [Morris, 2006]. For example, Litvak's halving method for pool testing involves dividing the population into pools, testing each pool, and if a pool tests positive, splitting it into two halves and repeating the process until individual positive cases are identified, minimizing the number of tests required. [Litvak, 1994]. Some extensions which can leverage *a priori* knowledge of some heterogeneity in the population [McMahan, 2012; Bilder, 2010; Black, 2012]; and combinatorial algorithms, such as [Li, 1962; Du, 2000]. Section 2 of this chapter will provide another example. The work of [Sterrett, 1957; Sobel, 1959] are important refinements of Dorfman's work. [Aldridge, 2019] provides a recent survey of the topic and [Du, 2000] is the reference book on the topic. Hwang's generalised binary-splitting algorithm [Hwang, 1972] works by performing a binary search on groups that test positive, and is a simple algorithm that finds a single defective in no more than the information-theoretic lower-bound number of tests. This has been improved by Allemann in 2013, with an algorithm performing $0.255d + \frac{1}{2}\log_2(d) + 5.5$ tests above the information lower bound when $n/d \geq 38$ and $d \geq 10$, where d is the quantity of positive individuals [Allemann, 2013].

¹United States Food and Drug Administration

²This function is defined for any $z \in \mathbb{C}$ as follows: $z = we^w \iff w = W(z)$.

It is important to distinguish between two types of pool tests: *Adaptive tests* where the tested samples depend on previously tested ones and *Non-adaptive tests*, where all the tests are planned in advance. All the testing strategies discussed so far are *adaptive*, in the sense that they may retest individuals based on the result of previous tests. The search for efficient and near-optimal *non-adaptive* tests is still a very open problem, motivated by the desire to perform tests in parallel and at scale, and can also be approached from an information-theoretic angle [Coja-Oghlan, 2019].

Pool tests are also classified as either *probabilistic* or *combinatorial*. In essence, probabilistic models assume a probability distribution and seek to optimize the average number of tests required to test all the patients. By opposition, combinatorial algorithms seek to minimize the worst-case number of tests when the probability distribution governing the infection is unknown.

1.2.1 Pool Testing and Shannon Entropy

The connection between pool testing and information theory was first made by Sobel and Groll in 1959 [Sobel, 1959; Aldridge, 2019]. We recall here their argument for the sake of clarity.

We consider a population, with each individual being either *positive* (+) or *negative* (−). We do not assume anything about what this labeling means medically. However we consider that it is possible to *pool-test* a group of individuals: by “mixing together” their samples, and testing the resulting mix, we obtain a certain outcome (+ or −). If any of the samples from this pool is +, then the outcome is +. Alternatively, if the pool-test outcome is −, then no individual from the tested group is +.

This method is well-known and practical within technical and ethical limits which are not within the scope of this thesis. We assume that the tests have negligible error rates. We also do not take into account dilution effects (i.e. the fact that the greater the pool is, the greater the chance is for false negative).

If the total population consists of n individuals, each carrying one information bit (whether they are + or −), then there is an n -bit string S describing the status of every individual. Testing one individual reveals the corresponding bit of S . Naturally, testing all individuals one by one reveals the complete string S . Trivially, any binary test (such as pool testing) reveals, again, at most one bit of information.

Shannon’s entropy measures $H(S)$, the amount of bits necessary to describe S . Therefore, any testing strategy providing complete and correct information on S must perform, on average, at least $H(S)$ tests. An “optimal” testing strategy would perform no more than $H(S)$ tests. A near-optimal strategy approaches this situation arbitrarily closely, within a ratio of $1 - \epsilon$. In this section we chose $\epsilon = 0.01$ to keep the exposition simple and concrete.

Advanced testing strategies better approaching the optimum exist, but their description is more intricate and would only result in marginal practical advantages over the strategies described in this thesis.

Finally, adaptive pool testing in the presence of a large percentage of positives is best done by individual testing, rather than by pooling. However, the positiveness probability making individual testing optimal is not known with certainty.

1.2.2 Related Research

Pool testing has already been used to screen large portions of the population in scarcely-infected areas (or as a best-effort measure, when test availability was low). Pool testing has been successfully used to identify viral diseases, such as HIV [Nguyen, 2019; Emmanuel, 1988], Zika [Bierlaire, 2017], Chlamydia [Currie, 2004],

Malaria [Taylor, 2010], and Influenza [Van, 2012]. In addition, pool testing has been suggested as a screening method for routine HCV, HBV, and HIV-1 PCR donors for a blood-bank [Roth, 1999].

During the Covid-19 pandemic, due to the urging need to test vast number of subjects, and with the uncertainty about prophylactic measures, the absence of efficient treatment, atop the threat on lives and hospital capacity, the idea of pool testing has become more and more appealing, to a point of becoming an area of intense interest. To this day, testing remains the only way to catch carriers of SARS-COV-2 at an early stage, which greatly increases the hope of limiting contagion, as well as successful recovery for the individual [Ghebreyesus, 2020].

While relatively efficient and precise tests were quickly developed, producing them at scale and distributing them was more of an issue. Test shortages [Erdman, 2020] and financial constraints made it necessary to reduce the costs associated with mass testing, which made pool testing an attractive solution [Hogan, 2020; Yelin, 2020; Sinnott-Armstrong, 2020; Shani-Narkiss, 2020; Torres, 2020; Eberhardt, 2020]. In several countries, such as Israel [Ben-Ami, 2020], Germany [Liu, 2020], South Korea [Chang-won, 2020], and some US³ [Ryan, 2020] and Indian⁴ states⁵ [India-Today, 2020] pool testing became the official testing procedure.

Field research focusing on reducing the number of tests [Farfan, 2020; Assad, 2020; Gollier, 2020] did not analyse prior information strategies but instead provided simulation (or small sample) results showing the benefits of pool testing. In most cases, the existing literature only uses pooling as a way to screen the infection in an emerging context, not as a precise approach to identify which individuals are infected and which are not.

We also note projects meant to reduce the amount of work required for pool testing: e.g. the Origami Assays [Woolf, 2020] Project, a collection of open source pool testing designs for standard 96 well plates. The Origami XL3 design tests 1120 patients in 94 assay wells.

Yelin et al. [Yelin, 2020] demonstrated that pool testing can be used effectively to identify one positive SARS-COV-2 result within 32 samples, and possibly within 64 samples if the cycles are amplified, with an estimated false-negative rate of 10%. [Täufer, 2020] uses a strategy consisting in running “cross batches”, where the same individuals are tested several times but in different pools, which eventually leads to positive sample identification. The resulting approach ends up using more tests overall (since it tests every individual more than once) than the strategy proposed in this work and does not exploit prior information. Similarly, Sinnott-Armstrong et al. [Sinnott-Armstrong, 2020] suggested to identify low-risk individuals (i.e. asymptomatic and mild cases) and to test them as a pool using a matrix-based method, so as to reduce the number of tests required by up to eight-fold, depending on the prevalence.

Albeit the obvious advantages in pool testing, this approach does have practical limitations that make its applicability sub-optimal: dilution while pooling makes detection in large pools difficult⁶; retesting individuals may be difficult, impossible or undesirable; the construction of the mixtures, which is done by technicians by hand, can be time-consuming and error-prone; error rates of actual tests may be sensitive to the marker’s concentration, and pooling may cause the result to be unexploitable due to large error margins.

This chapter is based on assumption that a successful emergency application of the refined pool testing

³e.g Nebraska.

⁴e.g. Uttar Pradesh, West Bengal, Punjab, Chhattisgarh, Maharashtra.

⁵Indian Council of Medical Research, *Advisory on feasibility of using pooled samples for molecular testing of Covid-19*, April 13, 2020.

⁶For SARS-COV-2, RT-PCR tests can work with pools of size about 32 [Yelin, 2020], which is far beyond what currently done in practice, around 5 to 10 [Hogan, 2020].

procedures described here would improve the Covid-19 testing capacity significantly.

2 Optimal Covid-19 Pool Testing with *A Priori* Information

Based on common work with Marc Beunardeau, Éric Brier, Noémie Cartier, Aisling Connolly, Nathanaël Courant, Rémi Géraud-Stewart and David Naccache.

This section deals with adaptive probabilistic tests; it departs from the above approaches by assuming the availability of extra information - the *a priori* probability that each given test is negative. In practice, we may either assume that such probabilities are given, estimated from patient trust metrics, or are learned from past Covid-19 tests. We assume in this work that these probabilities are known.

We show that it is possible to find positive samples in an optimal way, i.e., by performing on average the minimum number of tests. This turns out to be faster than blind divide-and-conquer testing in the vast majority of settings.

A concrete consequence of this research is the design of testing procedures that are faster and more cost-effective.

2.1 Intuition

Before introducing models and general formulae, let us provide the intuition behind our algorithms.

Let us begin by considering the very small case of two samples. These can be tested individually or together, in a pool. Individual Covid-19 testing claims a minimum two units of work—check one sample, then check the other. Pool-testing them requires a minimum of one Covid-19 test. If it is highly probable that both samples are negative, then pool testing is interesting: If both samples are indeed negative, we can make a conclusion after one test and halve the Covid-19 test’s cost. However, if that fails, we are nearly back to square one: One of these samples (at least) is positive, and we don’t know which one.

In this section, we identify *when* to check samples individually, and when to pool-test them instead—including all possible generalizations when there are more than two samples. We assume that the probability of a sample being positive is known to us in advance. The result is a testing “metaprocedure” that offers *the best alternative to sequential and individual testing*.

To demonstrate: the testing procedure that always works is to test every sample individually, one after the other: This gives the “naive procedure”, which always performs two Covid-19 tests, as illustrated in Figure 2.1. In this representation, the numbers in parentheses indicate which samples are being tested at any given point. The leaves indicate which samples are negative (denoted 1) or positive (denoted 0), for instance the leaf 01 indicates that only the second sample is healthy. Note that the order in which each element is tested does not matter: There are thus two equivalent naive procedures, namely the one represented in Figure 2.1, and the procedure obtained by switching the testing order of (1) and (2).

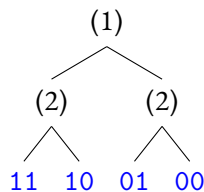


Figure 2.1: The “naive procedure” for $n = 2$ consists in testing each entity separately and sequentially.

Alternatively, we can leverage the possibility to test both samples together as the set $\{1, 2\}$. In this case, pooling the pair $\{1, 2\}$ must be the first step: Indeed, testing $\{1, 2\}$ after any other test would be redundant, and the definition of testing procedures prevents this from happening. If the test on $\{1, 2\}$ is negative, both samples are negative and the procedure immediately yields the outcome **11**. Otherwise, we must identify which of the samples 1 or 2 (or both) is responsible for the test's positiveness. There are thus two possible procedures, illustrated in Figure 2.2.

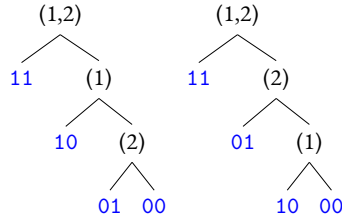


Figure 2.2: Two pooling testing procedures having $(1, 2)$ as root.

Intuitively, the possibility that this procedure terminates early indicates that, in some situations at least, only one test is performed, and is thus less costly than the naive procedure. However, in some situations up to three tests can be performed, in which case it is more costly than the naive procedure.

Concretely, we can compute how many Covid-19 tests are performed on average by each approach, depending on the probability x_1 that the first sample is positive, and x_2 that the second is positive. To each procedure, naive, pool-*left*, pool-*right*, we associate the following polynomials representing the expected stopping time:

- $L_{\text{naive}} = 2$
- $L_{\text{pool-left}} = (1 - x_1)(1 - x_2) + 2(1 - x_1)x_2 + 3x_1(1 - x_2) + 3x_1x_2$
- $L_{\text{pool-right}} = (1 - x_1)(1 - x_2) + 3(1 - x_1)x_2 + 2x_1(1 - x_2) + 3x_1x_2$

It is possible to see analytically which of these polynomials evaluates to the smallest value as a function of (x_1, x_2) . Looking at Figure 2.3, we use these expectations to define zones in $[0, 1]^2$ where each algorithm is optimal (i.e. the fastest on average). More precisely, the frontier between zones C and B has equation $x_1 = x_2$, the frontier between A and B has equation $x_2 = (x_1 - 1)/(x_1 - 2)$, the frontier between A and C has equation $x_2 = (2x_1 - 1)/(x_1 - 1)$, and the three zones meet at $\bar{x}_1 = \bar{x}_2 = (3 - \sqrt{5})/2$, a well-known cutoff value observed as early as 1960 [Ungar, 1960].

Having identified the zones, we can write an algorithm which, given x_1 and x_2 , identifies in which zone of Figure 2.3 (x_1, x_2) lies, and then apply the corresponding optimal testing sequence. In the specific case illustrated above, three algorithms out of three were needed to define the zones; however, for any larger scenario, we will see that only a very small portion of the potential algorithms will be carefully selected.

Our objective is to determine the zones, and the corresponding testing algorithms, for arbitrary n , so as to identify which samples in a set are negative and which are not, while minimizing the expected number of testing operations.

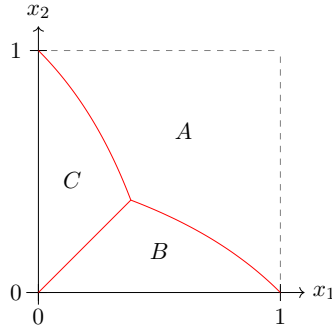


Figure 2.3: Optimality zones for $n = 2$. A : naive procedure; B : pooling procedure (right); C : pooling procedure (left).

2.2 Preliminaries

This section will formalize the notion of a testing procedure, and the cost thereof, so that the problem at hand can be mathematically described. We aim at the greatest generality, which leads us to introduce ‘and-tests’, a special case of which are samples that can be pool tested.

2.2.1 Testing Procedures

We consider a collection of n samples. Let $[n]$ denote $\{1, \dots, n\}$, and $\Omega = \mathcal{P}([n]) \setminus \{\emptyset\}$, where \mathcal{P} is the power set (ie. $\mathcal{P}(X)$ is the set of subsets of X).

Definition 2.1 (Test). A *test* is a function $\phi : \Omega \rightarrow \{0, 1\}$, that associates a bit to each subset of Ω .

We focus in this work on the following:

Definition 2.2 (And-Tests). An *and-test* $\phi : \Omega \rightarrow \{0, 1\}$ is a test satisfying the following property:

$$\forall T \in \Omega, \quad \phi(T) = \bigwedge_{t \in T} \phi(\{t\}).$$

In other terms, the result of an and-test on a set is exactly the logical and of the test results on individual members of that set.

Remark 1. Note that “or-tests”, where \wedge is replaced by \vee in the definition, are exactly dual to our setting. “xor-tests” can be defined as well but are not investigated here. Although theoretically interesting by their own right, we do not address the situation where both and-tests and or-tests are available, since we know of no concrete application where this is the case.

Elements of Ω can be interpreted as n -bit strings, with the natural interpretation where the i -th bit indicates whether i belongs to the subset. We call *selection* an element of Ω .

Definition 2.3 (Outcome). The *outcome* $F_\phi(T)$ of a test ϕ on $T \in \Omega$ is the string of individual test results:

$$F_\phi(T) = \{\phi(x), x \in T\} \in \{0, 1\}^n.$$

When $T = [n]$, F_ϕ will concisely denote $F_\phi([n])$.

Our purpose is to determine the outcome of a given test ϕ , by minimizing in the expected number of queries to ϕ . Note that this minimal expectation is trivially upper bounded by n .

Definition 2.4 (Splitting). Let $T \in \Omega$ be a selection and ϕ be a test. Let \mathcal{S} be a subset of Ω . The *positive part of \mathcal{S} with respect to T* , denoted \mathcal{S}_T^\top , is defined as the set

$$\mathcal{S}_T^\top = \{S \mid S \in \mathcal{S}, S \wedge T = T\}.$$

where the operation \wedge is performed element-wise. This splits \mathcal{S} into two. Similarly the complement $\mathcal{S}_T^\perp = \mathcal{S} - \mathcal{S}_T^\top$ is called the *negative part of \mathcal{S} with respect to T* .

We are interested in algorithms that find F_ϕ . More precisely, we focus our attention on the following:

Definition 2.5 (Testing procedure). A *testing procedure* is a binary tree \mathcal{T} with labeled nodes and leaves, such that:

- The leaves of \mathcal{T} are in one-to-one correspondence with Ω in string representation;
- Each node of \mathcal{T} which is not a leaf has exactly two children, (S_\perp, S_\top) , and is labeled (S, T) where $S \subseteq \Omega$ and $T \in \Omega$, such that

- $S_\perp \cap S_\top = \emptyset$
- $S_\perp \sqcup S_\top = S$
- $S_\perp = S_T^\perp$ and $S_\top = S_T^\top$.

Remark 2. It follows from the definition 2.5 that a testing procedure is always a *finite* binary tree, and that no useless calls to ϕ are performed. Indeed, doing so would result in an empty S for one of the children nodes. Furthermore, the root node has $S = \Omega$.

2.2.2 Interpreting and representing pooling procedures

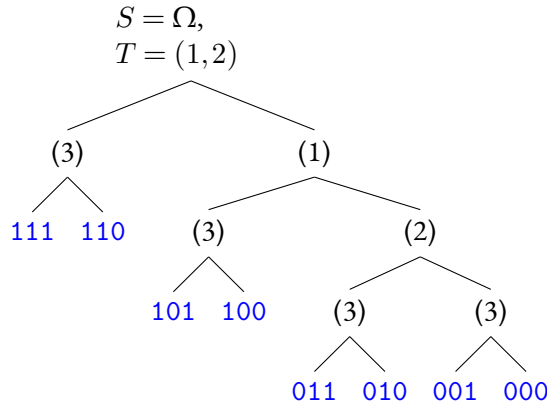


Figure 2.4: Graphical representation of a testing procedure. The collection is $[3] = \{1, 2, 3\}$, $\Omega = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$, the initial set of selections is $S = \Omega$. Only the T labels are written on nodes. Only the S labels are written for leaves.

Consider a testing procedure \mathcal{T} , defined as above. \mathcal{T} describes the following algorithm. At each node (S, T) , perform the test ϕ on the selection T of samples. If $\phi(T) = 0$, go to the left child; otherwise go to the right child. Note that at each node of a testing procedure, only one invocation of ϕ is performed.

The tree is finite and thus this algorithm reaches a leaf S_{final} in a finite number of steps. By design, $S_{\text{final}} = F_\phi$.

Remark 3. From now on, we will fix ϕ and assume it implicitly.

Remark 4. We represent a testing procedure graphically as follows: Nodes (in black) are labeled with T , whereas leaves (in blue) are labeled with S written as a binary string. This is illustrated in Figure 2.4 for $n = 3$.

This representation makes it easy to understand how the algorithm unfolds and what the outcomes are: Starting from the root, each node tells us which entity is tested. If the test is positive, the right branch is taken, otherwise the left branch is taken. Leaves indicate which samples tested positive and which samples tested negative from now on.

Remark 5. The successive steps of a testing procedure can be seen as imposing new logical constraints. These constraints ought to be satisfiable (otherwise one set S is empty in the tree, which cannot happen). The formula at a leaf is maximal in the sense that any additional constraint would make the formula unsatisfiable. This alternative description in terms of satisfiability of Boolean clauses is in fact strictly equivalent to the one that we gave.

In that case, T is understood as a conjunction $\bigwedge_{T[i]=1} t_i$, S is a proposition formed by a combination of terms t_i , connectors \vee and \wedge , and possibly \neg . The root has $S = \top$. The left child of a node labeled (T, S) is labeled $S_T^\perp = S \wedge (\neg T)$; while the right child is labeled $S_T^\top = S \wedge T$. At each node and leaf, S must be satisfiable.

2.2.3 Probabilities on trees

To determine how efficient any given testing procedure is, we need to introduce a probability measure, and a metric that counts how many calls to ϕ are performed.

We consider the discrete probability space (Ω, Pr) . The *expected value* of a random variable X is classically defined as:

$$\mathbb{E}[X] = \sum_{\omega \in \Omega} X(\omega) \text{Pr}(\omega)$$

Let \mathcal{T} a testing procedure, and let $S \in \Omega$ be one of its leaves. The *length* $\ell_{\mathcal{T}}(S)$ of \mathcal{T} over S is the distance on the tree from the root of \mathcal{T} to the leaf S . This corresponds to the number of tests required to find S if S is the outcome of ϕ . The *expected length* of a testing procedure \mathcal{T} is defined naturally as:

$$L_{\mathcal{T}} = \mathbb{E}[\ell_{\mathcal{T}}] = \sum_{\omega \in \Omega} \ell_{\mathcal{T}}(\omega) \text{Pr}(\omega)$$

It remains to specify the probabilities $\text{Pr}(\omega)$, i.e. for any given binary string ω , the probability that ω is the outcome.

If the different tests are independent, we can answer this question directly with the following result:

Lemma 1. Assume that the events “ $\phi(\{i\}) = 1$ ” and “ $\phi(\{j\}) = 1$ ” are independent for $i \neq j$. Then, $\forall \omega \in \Omega$,

$\Pr(\omega)$ can be written as a product of monomials of degree 1 in x_1, \dots, x_n , where

$$x_i = \Pr(\phi(\{i\}) = 1) = \Pr(i\text{-th bit of } \omega = 1).$$

Thus $L_{\mathcal{T}}$ is a multivariate polynomial of degree n with integer coefficients.

In fact, or-tests provide inherently independent tests. Therefore we will safely assume that the independence assumption holds⁷.

Example 2.1. Let $n = 5$ and $\omega = 11101$, then $\Pr(\omega) = x_1x_2x_3(1-x_4)x_5$.

Remark 6. $L_{\mathcal{T}}$ is uniquely determined as a polynomial by the integer vector of length 2^n defined by all its lengths: $\ell(\mathcal{T}) = (\ell_{\mathcal{T}}(0\dots 0), \dots, \ell_{\mathcal{T}}(1\dots 1))$.

2.3 Optimal Pool Tests

We have now introduced everything necessary to state our goal mathematically. Our objective is to identify the best performing testing procedure \mathcal{T} (i.e. having the smallest $L_{\mathcal{T}}$) in a given situation, i.e. knowing $\Pr(\omega)$ for all $\omega \in \Omega$.

2.4 Generating all procedures

We can now explain how to generate all the testing procedures for a given $n \geq 2$.

One straightforward method is to implement a generation algorithm based on the definition of a testing procedure. Algorithm 1 does so recursively by using a co-routine. The complete list of testing procedures is recovered by calling `FindProcedure($\Omega, \Omega \setminus \{\emptyset\}$)`.

⁷In practice, it may be that this assumption is too optimistic: members of a given household or community are more likely to have similar results, see [Sinnott-Armstrong, 2020]. Choosing test subjects at random rather than geographically makes this assumption more likely to be verified.

Algorithm 1: FindProcedure

Input: $S \in \Omega, C \in \Omega$.

Output: A binary tree.

- if $|S| == 1$ then return S
- $S'_\perp = S'_\top = C' = \emptyset$
- for each $c \in C$
 - $S_\perp = S_c^\perp$
 - $S_\top = S_c^\top$
 - if $S_\perp \notin S'_\perp$ and $S_\top \notin S'_\top$
 - $S'_\perp = S'_\perp \cup \{S_\perp\}$
 - $S'_\top = S'_\top \cup \{S_\top\}$
 - $C' = C' \cup \{c\}$
- for $i \in \{1, \dots, |C'|\}$
 - $\overline{C} = C - C'[i]$
 - for each $\mathcal{T}_\perp \in \text{FindProcedure}(S'_\perp[i], \overline{C})$
 - for each $\mathcal{T}_\top \in \text{FindProcedure}(S'_\top[i], \overline{C})$
 - yield $(C'[i], \mathcal{T}_\perp, \mathcal{T}_\top)$

We implemented this algorithm (in Python) The result of testing procedure generations for small values of n is summarized in Table 2.1. The number of possible testing procedures grows very quickly with n .

Table 2.1: Generation results for some small n

n	Number of procedures	Time
1	1	0
2	4	~ 0
3	312	~ 0
4	36585024	~ 30 mn

An informal description of Algorithm 1 is the following. Assuming that you have an unfinished procedure (i.e. nodes at the end of branches are not all leaves). For those nodes S , compute for each T the sets S_T^\top and S_T^\perp . If either is empty, abort. Otherwise, create a new (unfinished) procedure, and launch recursively on nodes (not on leaves, which are such that S has size 1).

Algorithm 1 terminates because it only calls itself with strictly smaller arguments. We will discuss this algorithm further after describing some properties of the problem at hand.

2.5 Metaprocedures

Once the optimality zones and the corresponding testing procedures, have been identified, it is easy to write an algorithm which calls the best testing procedure in every scenario. At first sight, it may seem that nothing is

gained from doing so — but as it turns out that only a handful of procedures need to be implemented.

This construction is captured by the following definition:

Definition 2.6 (Metaprocedure). A *metaprocedure* \mathcal{M} is a collection of pairs (Z_i, \mathcal{T}_i) such that:

- $Z_i \subseteq [0, 1]^n$, $Z_i \cap Z_j = \emptyset$ whenever $i \neq j$ and $\bigsqcup_i Z_i = [0, 1]^n$.
- \mathcal{T}_i is a testing procedure and for any testing procedure \mathcal{T} ,

$$\forall x \in Z_i, \quad L_{\mathcal{T}_i}(x) \leq L_{\mathcal{T}}(x).$$

A metaprocedure is interpreted as follows: Given $x \in [0, 1]^n$ find the unique Z_i that contains x and run the corresponding testing procedure \mathcal{T}_i . We extend the notion of expected length accordingly: $L_{\mathcal{M}} = \min_i L_{\mathcal{T}_i} \leq n$.

One way to find the metaprocedure for n , is to enumerate all the testing procedures using Algorithm 1, compute all expected lengths $L_{\mathcal{T}}$ from the tree structure, and solve polynomial inequalities.

Surprisingly, a vast majority of the procedures generated are nowhere optimal: This is illustrated in Table 2.2. Furthermore, amongst the remaining procedures, there is a high level of symmetry. For instance, in the case $n = 3$, eight procedures appear 6 times, one procedure appears 3 times, and one procedure appears once. The only difference between the occurrences of these procedures — which explains why we count them several times — is the action of the symmetric group S_6 on the cube (see section 2.11 for a complete description).

The metaprocedure for $n = 3$ cuts the unit cube into 52 zones, which correspond to a highly symmetric and intricate partition, as illustrated in Figures 2.5, 2.6, and 2.7. An STL model was constructed and is available upon request.

The large number of suboptimal procedures shows that the generate-then-eliminate approach quickly runs out of steam: Generating all procedures for $n = 6$ seems out of reach with Algorithm 1⁸. The number of zones, which corresponds to the number of procedures that are optimal in some situation, is on the contrary very reasonable.

Lemma 2 (Number of naive procedures). Let $n \geq 1$, then there are

$$P(n) = \prod_{k=1}^n k^{2^{n-k}}$$

equivalent naive procedures.

Proof. By induction on n : There are $(n + 1)$ choices of a root node, $P(n)$ choices for the left child, and $P(n)$ choices for the right child. This gives the recurrence $P(n + 1) = (n + 1)P(n)^2$, hence the result. \square

This number grows rapidly and constitutes a lower bound for the total number of procedures (e.g. for $n = 8$ we have $P(n) > 2^{184}$). On the other hand, the naive procedure is the one with maximal multiplicity, which yields a crude upper bound $C_{2k}P(n)$ on the number of procedures, where C_t is the t -th Catalan number defined by:

⁸ $n = 6$ is still very far from the current SARS-COV-2 test pooling capacity of $n = 32$ or $n = 64$ mentioned in the introduction.

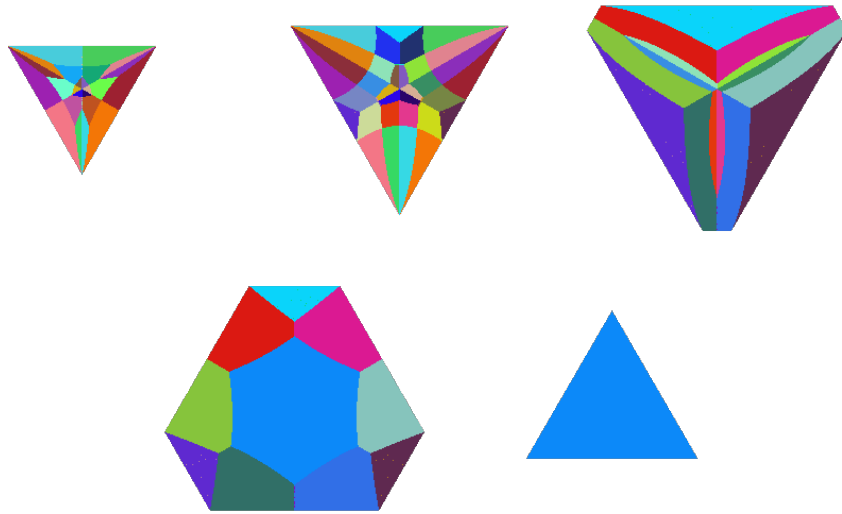


Figure 2.5: Slices of the cube decomposition for the $n = 3$ metaprocedure, each colour corresponds to a different strategy, which is optimal at this position. The slices are taken orthogonally to the cube's main diagonal, with the origin at the center of each picture. Each color corresponds to a procedure. The symmetries are particularly visible.

$$C_t = \frac{1}{t+1} \binom{2t}{t} \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}$$

n	Number of procedures	Zones
1	1	1
2	4	3
3	312	52
4	36585024	181
5	$8.926 \cdot 10^{20}$?
6	$2.242 \cdot 10^{55}$?

Table 2.2: Procedures and metaprocedures for some values of n . The number of zones for $n = 5$ and 6 cannot be determined in a reasonable time with the generate-then-eliminate approach.

The zones can be determined by sampling precisely enough the probability space. Simple arguments about the regularity of polynomials guarantee that this procedure succeeds, when working with infinite numerical precision. In practice, although working with infinite precision is feasible (using rationals), we opted for floating-point numbers, which are faster. The consequence is that sometimes this lack of precision results in incorrect results on the zone borders — however this is easily improved by increasing the precision or checking manually that there is no sub-zone near the borders.

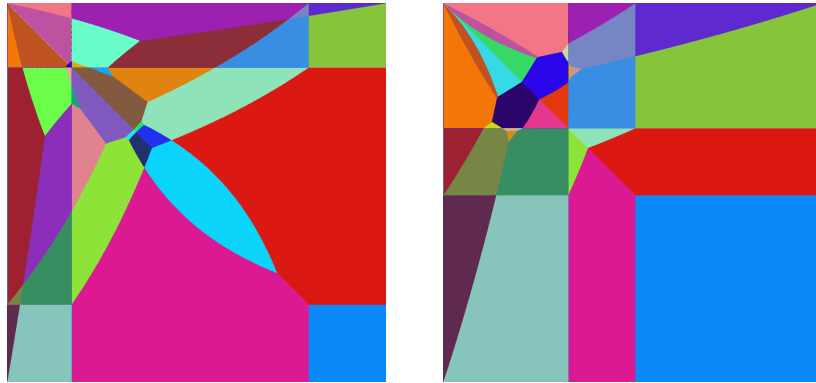


Figure 2.6: Slices through the cube at the $z = 0.17$ (left) and the $z = 0.33$ (right) planes, showing the metaprocedure's rich structure. Each colour corresponds to a different strategy, which is optimal at this position. The origin is at the top left.

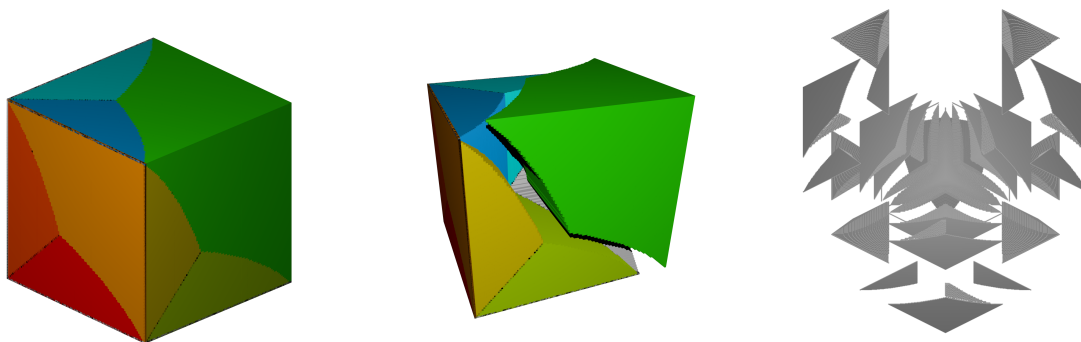


Figure 2.7: A 3D visualisation of the cube. Left: exterior, where it is visible that each face has the same decomposition as the 2D problem; Middle: with the naive algorithm region slightly removed, showing that it accounts for slightly less than half of the total volume; Right: exploded view of the 52 substructures (looking from $(-1, -1, -1)$).

2.6 Pruning the Generation Tree

We now focus on some of the properties exhibited by testing procedures, which allows a better understanding of the problem and interesting optimizations. This in effect can be used to prune early the generation of procedures, and write them in a more compact way by leveraging symmetries. We consider in this section a testing procedure \mathcal{T} .

Lemma 3. Let B_0 and B_1 be two binary strings of size n , that only differ by one bit (i.e. $B_0[i] = 0$ and $B_1[i] = 1$ for some i). Then $\ell_{\mathcal{T}}(B_0) \leq \ell_{\mathcal{T}}(B_1)$.

Proof. First notice that for all T, T' , and $b, b' \in \{\top, \perp\}$ we have $(S_T^b)_{T'}^{b'} = (S_{T'}^{b'})_T^b$. We will denote both by $S_{TT'}^{bb'}$.

We have the following : If there exists k, T_1, \dots, T_k , and β_1, \dots, β_k such that

$$(\Omega)_{T_1 \dots T_k}^{\beta_1 \dots \beta_k} = \{B_1\}$$

then there exists $i \leq k$ such that

$$(\Omega)_{T_1 \dots T_i \dots T_k}^{\beta_1 \dots \neg \beta_i \dots \beta_k} = \{B_0\}$$

Indeed there exists $i \leq k$ such that $\beta_i = \top$ and $T_i = \{i_0\} \cup E$ where for all j in E , $B_0[j] = B_1[j] = 0$. This yields

$$(\Omega)_{T_1 \dots T_{i-1} T_{i+1} \dots T_k}^{\beta_1 \dots \beta_{i-1} \neg \beta_{i+1} \dots \beta_k} = \{B_0, B_1\}$$

and the result follows. \square

Remark 7. Proposition 3 indicates that testing procedures are, in general, unbalanced binary trees: The only balanced procedure being the naive one.

Lemma 4. If \mathcal{N} is the naive procedure, then for any testing procedure \mathcal{T} and for all x_1, \dots, x_n such that $x_i > \frac{1}{2}$,

$$L_{\mathcal{N}}(x_1, \dots, x_n) \leq L_{\mathcal{T}}(x_1, \dots, x_n).$$

In other terms $\{\forall i \in [n], \frac{1}{2} \leq x_i \leq 1\}$ is contained in the naive procedure's optimality zone.

Proof. An immediate corollary of Proposition 3 is that for all $i \in [n]$, we have $\partial_{x_i} L_{\mathcal{T}}(x_1, \dots, x_n) \geq 0$, where ∂_{x_i} indicates the derivative with respect to the variable x_i . Since the naive procedure has a constant length, it suffices to show that it is optimal at the point $\{\frac{1}{2}, \dots, \frac{1}{2}\}$. Evaluating the length polynomials at this point gives

$$L_{\mathcal{T}}\left(\frac{1}{2}, \dots, \frac{1}{2}\right) = \frac{1}{2^n} \sum_{\omega \in \Omega} \ell_{\mathcal{T}}(\omega) = \int_{[0,1]^n} L_{\mathcal{T}} dx.$$

Now remember that the naive procedure gives the only perfect tree. It suffices to show that unbalancing this tree in any way results in a longer sum in the equation above. Indeed, to unbalance the tree one needs to:

- Remove two bottom-level leaves, turning their root node into a leaf
- Turn one bottom-level leaf into a node
- Attach two nodes to this newly-created leaf

The total impact on the sum of lengths is $+1$. Hence the naive algorithm is minimal at $\{\frac{1}{2}, \dots, \frac{1}{2}\}$, and therefore, in the region $\{\forall i \in [n], \frac{1}{2} \leq x_i \leq 1\}$. \square

Remark 8. This also shows that if we assume that the probabilities are supposed to be uniform (ie. we assume no *a priori* knowledge) the optimal procedure is the naive one. Therefore we can see that the gain for $n = 3$ is approximately 0.34 since the optimal procedure in average gives 2.66. In percentage the gain is 15%. If the probabilities are very low we have a gain of almost 2, which is 3 times faster. As expected, it is much more interesting if we think that the samples have a good chance to be negative, which is the case in most real life scenarii.

Lemma 5. If the root has a test of cardinality one, then the same algorithms starting at both sons have same expected stopping time. This applies if the next test is also of cardinal one.

Proof. Without loss of generality we can assume that the test is $\{1\}$. We have $\{0, 1\}_{\{1\}}^{n\top} = \{0b_2 \cdots b_n | b_2 \cdots b_n \in \{0, 1\}^{n-1}\}$ and $\{0, 1\}_{\{1\}}^{n\perp} = \{1b_2 \cdots b_n | b_2 \cdots b_n \in \{0, 1\}^{n-1}\}$. A test T that does not test 1 applied on those sets will give the same split for both, and the probability that the test answers yes or no is the same. This is also true for the sets and the tests T such that i is not in T for i in $\{1, \dots, k\}$. $\{0^k b_2 \cdots b_n | b_2 \cdots b_n \in \{0, 1\}^{n-k}\}$ and $\{01^k b_2 \cdots b_n | b_2 \cdots b_n \in \{0, 1\}^{n-k}\}$. A test T such that there exists i in $T \setminus \{1, \dots, k\}$ brings no information for the set of possibilities $\{01^k b_2 \cdots b_n | b_2 \cdots b_n \in \{0, 1\}^{n-k}\}$, but testing this i is useless for the set $\{0^k b_2 \cdots b_n | b_2 \cdots b_n \in \{0, 1\}^{n-k}\}$. So we can apply the test $T - \{1, \dots, k\}$. \square

Corollary 1. If the root has a test of cardinal one, then an optimal algorithm can always apply the same test for the right and left child. If this test is also of cardinal one then the property is still true.

This result helps in identifying redundant descriptions of testing procedures, and can be used to narrow down the generation, by skipping over obvious symmetries of the naive procedure (see Figure 2.8).

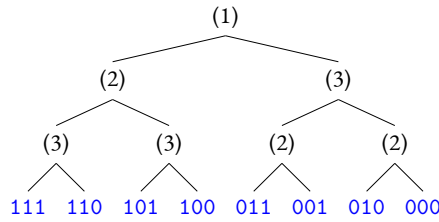


Figure 2.8: Naive algorithm, where the order of tests are unimportant in the left and right branches.

To further accelerate generation we can only keep one representative of each algorithms that have the same expected length for all x_i .

Lemma 6. If a node labeled T_1 has two children that are *both* labeled T_2 , then we can interchange T_1 and T_2 without changing the testing procedure's expected length.

Yet another simple observation allows to reduce the set of subsets T at each step:

Lemma 7. Consider a node labeled (T, \mathcal{S}) . Assume that there is $i \in [n]$ such that, for all S in \mathcal{S} , $i \notin S$. Then we can replace T by $T \cup \{i\}$.

Proof. We can easily see that $S_T^\top = S_{T \cup \{i\}}^\top$ and $S_T^\perp = S_{T \cup \{i\}}^\perp$. \square

Finally we can leverage the fact that the solutions exhibit symmetries, which provides both a compact encoding of testing procedures, and an appreciable reduction in problem size.

Lemma 8. Let $\sigma \in \mathfrak{S}_n$ be a permutation on n elements. If we apply σ to each node and leaf of \mathcal{T} , which we can write $\sigma(\mathcal{T})$, then

$$L_{\sigma(\mathcal{T})}(x_1, \dots, x_n) = L_{\mathcal{T}}(\sigma(x_1, \dots, x_n)).$$

Proof. Note that for any $S \in \Omega$ and $T \in \Omega \setminus \{\emptyset\}$ we have $\sigma(S_T^\top) = S_{\sigma(T)}^\top$ and $\sigma(S_T^\perp) = S_{\sigma(T)}^\perp$, where σ operates on each binary string. It follows that for any leaf S , $\ell_{\mathcal{T}}(S)$ becomes $\ell_{\mathcal{T}}(\sigma(S))$ under the action of σ , hence the result. \square

Lemma 9. Let S be a simplex of the hypercube, \mathcal{T} a procedure, $E = \{\sigma(\mathcal{T}) \mid \sigma \in \mathfrak{S}_n\}$, then there exists \mathcal{T}_0 in E , such that for all x in S , \mathcal{T}_1 in E we have

$$L_{\mathcal{T}_0}(x) \leq L_{\mathcal{T}_1}(x).$$

Moreover we have for all σ in \mathfrak{S}_n , x in $\sigma(S)$, \mathcal{T}_1 in E

$$L_{\sigma(\mathcal{T}_0)}(x) \leq L_{\mathcal{T}_1}(x).$$

Remark 9. The last two propositions allow us to solve the problem on a simplex of the hypercube (of volume $1/n!$) such as $\{p_1, \dots, p_n \mid 1 \geq p_1 \geq \dots \geq p_n \geq 0\}$.

2.7 Best Testing Procedure at a Point

We examine the following problem: Find the testing procedure \mathcal{T} for a given $k \leq n$, $(p_{i_1}, \dots, p_{i_k}) \in [0, 1]^n$, and a selection $P \subseteq 2^{[k]}$ that satisfies:

- $\mathcal{S}_{\mathcal{T}} = P$,
- \mathcal{T} is optimal at point $(p_{i_1}, \dots, p_{i_k})$

This can be computed using a dynamic programming technique, by examining the outcome of each possible test that is the root node of the testing procedure \mathcal{T} , which gives Algorithm 2.

The same dynamic programming algorithm can also be used to compute the number of testing procedures (including those leading to duplicate polynomials) that exist in a given dimension. It is actually even easier, since there is a huge number of symmetries that can be exploited to count.⁹

Definition 2.7 (Decided point). We say that x is a *decided point* for \mathcal{S} a set of selections if either of the following is true:

- $x \in S$ for all $S \in \mathcal{S}$
- $x \notin S$ for all $S \in \mathcal{S}$

In the first case, we will say that x is a *positive decided point*, and a *negative decided point* in the second case.

We denote by $\mathcal{D}_{\mathcal{S}}^+$ the set of positive decided points of \mathcal{S} , $\mathcal{D}_{\mathcal{S}}^-$ its set of negative decided points, and $\mathcal{D}_{\mathcal{S}} = \mathcal{D}_{\mathcal{S}}^+ \cup \mathcal{D}_{\mathcal{S}}^-$ its set of decided points.

⁹Indeed, we can apply the algorithm to an even higher dimension than our solution to the given point problem.

Algorithm 2: FindOptimal

Input: $k \geq 0, (p_1, \dots, p_k) \in [0, 1]^k, \mathcal{S} \subset 2^{[k]}$.

Output: The optimal testing procedure \mathcal{T} at point (p_1, \dots, p_k) which satisfies $\mathcal{S}_{\mathcal{T}} = \mathcal{S}$.

1. if $k == 0$ then return the naive algorithm
2. if $|\mathcal{D}_{\mathcal{S}}| > 0$
3. $U \leftarrow \{u_1, \dots, u_{\ell}\} = [k] \setminus \mathcal{D}_{\mathcal{S}}$
4. $\mathcal{R} \leftarrow \{\{r_1, \dots, r_p\} \mid \{u_{r_1}, \dots, u_{r_p}\} \cup \mathcal{D}_{\mathcal{S}}^+\}$
5. $\mathcal{T} \leftarrow \text{FindOptimal}(\ell, (p_{u_1}, \dots, p_{u_{\ell}}), \mathcal{R})$
6. replace $\{t_1, \dots, t_r\}$ by $\{u_{t_1}, \dots, u_{t_r}\}$ in \mathcal{T}
7. replace $\{\ell_1, \dots, \ell_r\}$ by $\{u_{\ell_1}, \dots, u_{\ell_r}\} \cup \mathcal{D}_{\mathcal{S}}^+$ in \mathcal{T}
8. else
9. $W \leftarrow \emptyset$
10. for each $T \subseteq [k]$
11. $\mathcal{S}_{\perp} \leftarrow \mathcal{S}_T^{\perp}$
12. $\mathcal{S}_{\top} \leftarrow \mathcal{S}_T^{\top}$
13. if $\mathcal{S}_{\perp} = \emptyset$ or $\mathcal{S}_{\top} = \emptyset$ then continue
14. $\mathcal{T}_{\perp} \leftarrow \text{FindOptimal}(k, (p_1, \dots, p_k), \mathcal{S}_{\perp})$
15. $\mathcal{T}_{\top} \leftarrow \text{FindOptimal}(k, (p_1, \dots, p_k), \mathcal{S}_{\top})$
16. $W \leftarrow W \cup \{(\mathcal{T}, \mathcal{T}_{\perp}, \mathcal{T}_{\top})\}$
17. return the best algorithm in W at point (p_1, \dots, p_n)

Counting the number of algorithms in a given dimension works the same way; the only difference is that there is no need to look at the probabilities, and thus, the resulting Algorithm 3 does fewer recursive calls and is faster.¹⁰

¹⁰We are not aware of a closed-form formula providing the same values as this algorithm.

Algorithm 3: CountAlgorithms

Input: $k \geq 0, \mathcal{S} \subset 2^{[k]}$.

Output: The number of testing procedures which satisfy $\mathcal{S}_{\mathcal{T}} = \mathcal{S}$.

1. if $k == 0$ then return 1
2. if $|\mathcal{D}_{\mathcal{S}}| > 0$
3. $U \leftarrow \{u_1, \dots, u_{\ell}\} = [k] \setminus \mathcal{D}_{\mathcal{S}}$
4. $\mathcal{R} = \{\{r_1, \dots, r_p\} \mid \{u_{r_1}, \dots, u_{r_p}\} \cup \mathcal{D}_{\mathcal{S}}^+\}$
5. return CountAlgorithms(ℓ, \mathcal{R})
6. $c \leftarrow 0$
7. for each $T \subseteq [k]$
8. $\mathcal{S}_{\perp} \leftarrow \mathcal{S}_T^{\perp}$
9. $\mathcal{S}_{\top} \leftarrow \mathcal{S}_T^{\top}$
10. if $\mathcal{S}_{\perp} = \emptyset$ or $\mathcal{S}_{\top} = \emptyset$ then continue
11. $c_{\perp} \leftarrow \text{CountAlgorithms}(k, (p_1, \dots, p_k), \mathcal{S}_{\perp})$
12. $c_{\top} \leftarrow \text{CountAlgorithms}(k, (p_1, \dots, p_k), \mathcal{S}_{\top})$
13. $c \leftarrow c + c_{\top} c_{\perp}$
14. return c

2.8 Enumerating procedures for $n = 3$

All the procedures for $n = 3$ that are optimal at some point, up to symmetries, are represented in Figure 2.9.

2.9 Conclusion and Open Questions

We have introduced the question of optimal pool testing with *a priori* probabilities, where one is given a set of samples and must determine in the least average number of operations which samples are negative, and which are not. We formalized this problem and pointed out several interesting combinatorial and algebraic properties that speed up the computation of an optimal sequence of operations — which we call a *metaprocedure*. We determined the exact solution for up to 4 samples.

For larger values, our approach requires too many computation to be tractable, and thus an exact solution is out of reach; however we gave several heuristic algorithms that scale well. We showed that these heuristics are sub-optimal in all cases, but they always do better than standard screening. The existence of a polynomial-time algorithm that finds optimal metaprocedures for large value of n is an open question — although there is probably more hope in finding better heuristics. An alternative would be to modify our generation algorithm to kill branches when the resulting expected lengths are all worse than some already-known procedure.

Once the metaprocedure for a given n is known, which only needs to be computed once, implementation is straightforward and only invokes a handful of (automatically generated) cases.

Finally, in our model we do not consider false positives and false negatives. In other words, tests are assumed

to be 100% accurate. Integrating in the model false positive and false negative probabilities is an interesting research challenge.

Besides the performance gain resulting from implementing metaprocedures for sample testing, the very general framework allows for applications in medical and engineering tests.

2.10 Approximation Heuristics

The approach consisting in generating many candidates, only to select a few, is wasteful. In fact, for large values of n (even from 10), generating all the candidates is beyond reach, despite the optimizations we described.

Instead, one would like to obtain the optimal testing procedure *directly*. It is a somewhat simpler problem, and we can find the solution by improving on our generation-then-selection algorithm (see subsection 2.7). However if we wish to address larger values of n , we must relax the constraints and use the heuristic algorithms described below, which achieve near-optimal results. This would be useful in real life scenarii for Covid-19 tests since we would like to test hundreds or more samples to have real gain.

2.10.1 Information-Based Heuristic

We first associate a “cost” to each outcome S , and set of outcomes \mathcal{S} :

$$\begin{aligned} \text{cost}(S, \mathcal{S}) &= f(S, \mathcal{S}) + g(S, \mathcal{S}) \\ f(S, \mathcal{S}) &= \#\{i \in [n] \text{ s.t. } s[i] = 1 \text{ and } \exists S' \in \mathcal{S}, S'[i] = 0\} \\ g(S, \mathcal{S}) &= \begin{cases} 1 & \text{if } \exists i \in \{i \in [n] \text{ s.t. } S[i] = 0\}, \exists S' \in \mathcal{S}, S'[i] = 1 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

This function approximates the smallest integer n such that there exists n calls to ϕ with arguments T_1, \dots, T_n , and β_1, \dots, β_n in $\{\perp, \top\}$ with $\mathcal{S}_{T_1, \dots, T_n}^{\beta_1, \dots, \beta_n} = \{S\}$. This function is used to define a “gain” function evaluating how much information is gathered when performing a test knowing the set of outcomes:

$$\text{gain}(T, \mathcal{S}) = \sum_{S \in \mathcal{S}_T^\top} \left(1 - \frac{\text{cost}(S, \mathcal{S}_T^\top)}{\text{cost}(S, \mathcal{S})}\right) \Pr(S) + \sum_{S \in \mathcal{S}_T^\perp} \left(1 - \frac{\text{cost}(S, \mathcal{S}_T^\perp)}{\text{cost}(S, \mathcal{S})}\right) \Pr(S)$$

Intuitively, we give higher gains to subsets T on which testing gives more information. Note that, if a call to ϕ does not give any information (i.e. \mathcal{S}_T^\top or \mathcal{S}_T^\perp is empty), then $\text{gain}(T, \mathcal{S}) = 0$.

This heuristic provides us with a greedy algorithm that is straightforward to implement. For given values x_1, \dots, x_n we thus obtain a testing procedure \mathcal{T}_H .

2.10.1.1 Testing the heuristic. We compared numerically \mathcal{T}_H to the metaprocedure found by exhaustion in the case $n = 3$. The comparison consists in sampling points at random, and computing the sample mean of each algorithm’s length on this input. The heuristic procedure gives a mean of 2.666, which under-performs the optimal procedure (2.661) by only 1%.

2.10.1.2 Counter-example to optimality. In some cases, the heuristic procedure behaves very differently from the metaprocedure. For instance, for $n = 3$, $x_1 = 0.01$, $x_2 = 0.17$, $x_3 = 0.51$, the metaprocedure yields

a tree which has an expected length of 1.889. The heuristic however produces a tree which has expected length 1.96. Both trees are represented in Figure 2.10.

Beyond their different lengths, the main difference between the two procedures of Figure 2.10 begin at the third node. At that node the set S is the same, namely $\{010, 011, 100, 101, 110, 111\}$, but the two procedures settle for a different T : The metaprocedure splits S , with $T = \{1, 3\}$, into $S_T^\perp = \{010\}$ and $S_T^\top = \{011, 100, 101, 110, 111\}$; while the heuristic chooses $T = \{1\}$ instead, and gets $S_T^\perp = \{010, 011\}$ and $S_T^\top = \{100, 101, 110, 111\}$.

To understand this difference, first notice that besides 010 and 011, all leaves are associated to a very low probability. The heuristic fails to capture that by choosing $T = \{1, 3\}$ early, it could later rule out the leaf 010 in one step and 011 in two. There does not seem to be a simple greedy way to detect this early on.

2.10.2 Pairing Heuristic

Another approach is to use small metaprocedures on subsets of the complete problem. Concretely, given n samples to test, place them at random into k -tuples (from some small value k , e.g. 5). Then apply the k -metaprocedure on these tuples. While sub-optimal, this approach does not yield worst results than the naive procedure.

In cases where it makes sense to assume that all the x_i are equal, then we may even recursively use the metaprocedures, i.e. the metaprocedures to be run are themselves placed into k -tuples, etc. Using lazy evaluation, only the necessary tests are performed.

2.11 Equivalences and Symmetries for $n = 3$

A procedure can undergo a transformation that leaves its expected length unchanged. Such transformations are called *equivalences*. On the other hand, Lemma 8 shows that some transformations operate a permutation σ on the variables x_i — such transformations are called *symmetries*.

Equivalences and symmetries are responsible for a large part of the combinatorial explosion observed when generating all procedures. By focusing on procedures up to symmetry, we can thus describe the complete set in a more compact way and attempt a first classification.

In the following representations (Figures 2.11, 2.12, and 2.13), blue indicates a fixed part, and red indicate a part undergoing some permutation. Double-headed arrows indicate that swapping nodes is possible. The number of symmetries obtained by such an operation is indicated under the curly brace below.

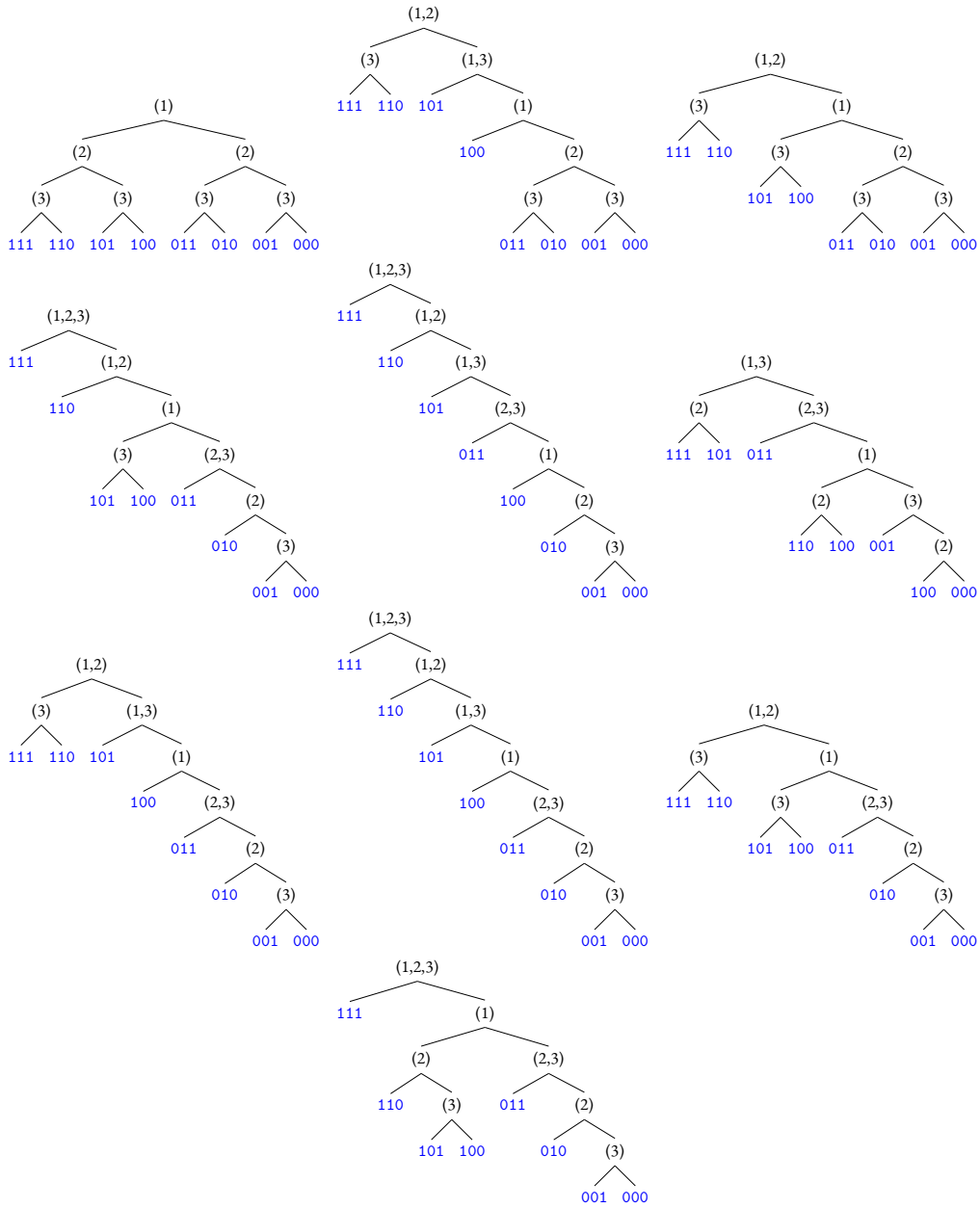


Figure 2.9: Optimal procedures (without permutations) for each zone when $n = 3$.

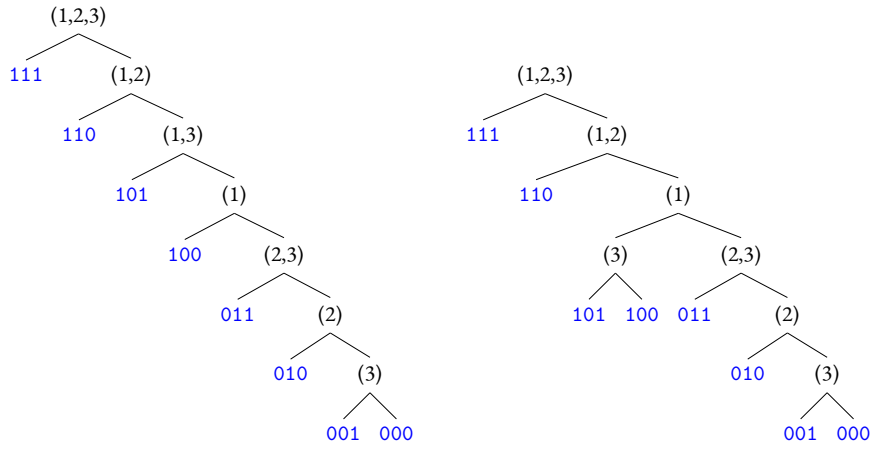


Figure 2.10: The optimal metaprocedure tree (left), and heuristic metaprocedure (right) for the same point $x = (0.01, 0.17, 0.51)$. The optimal procedure has expected length 1.889, as compared to 1.96 for the heuristic procedure.

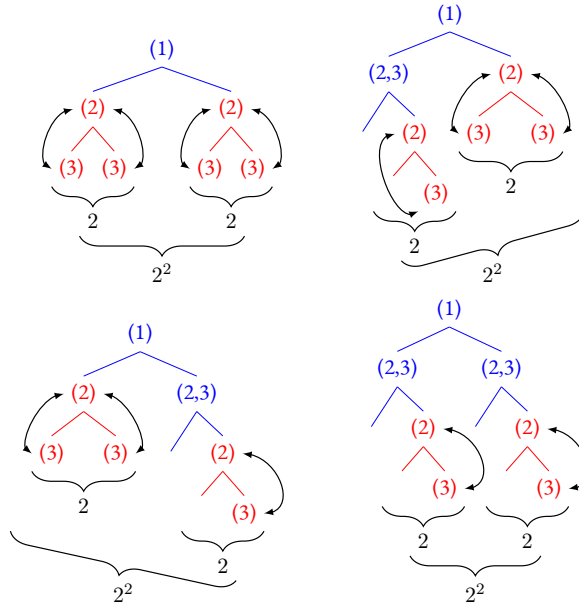


Figure 2.11: Trees representation with a grouping by one element on the root. For a fixed element, we have 2^2 possible permutations. Since we have 4 patterns, we get $2^2 \times 4$ possible permutations for one grouping. Hence, we finally have $2^2 \times 4 \times 3$ for all possible groupings by one element.

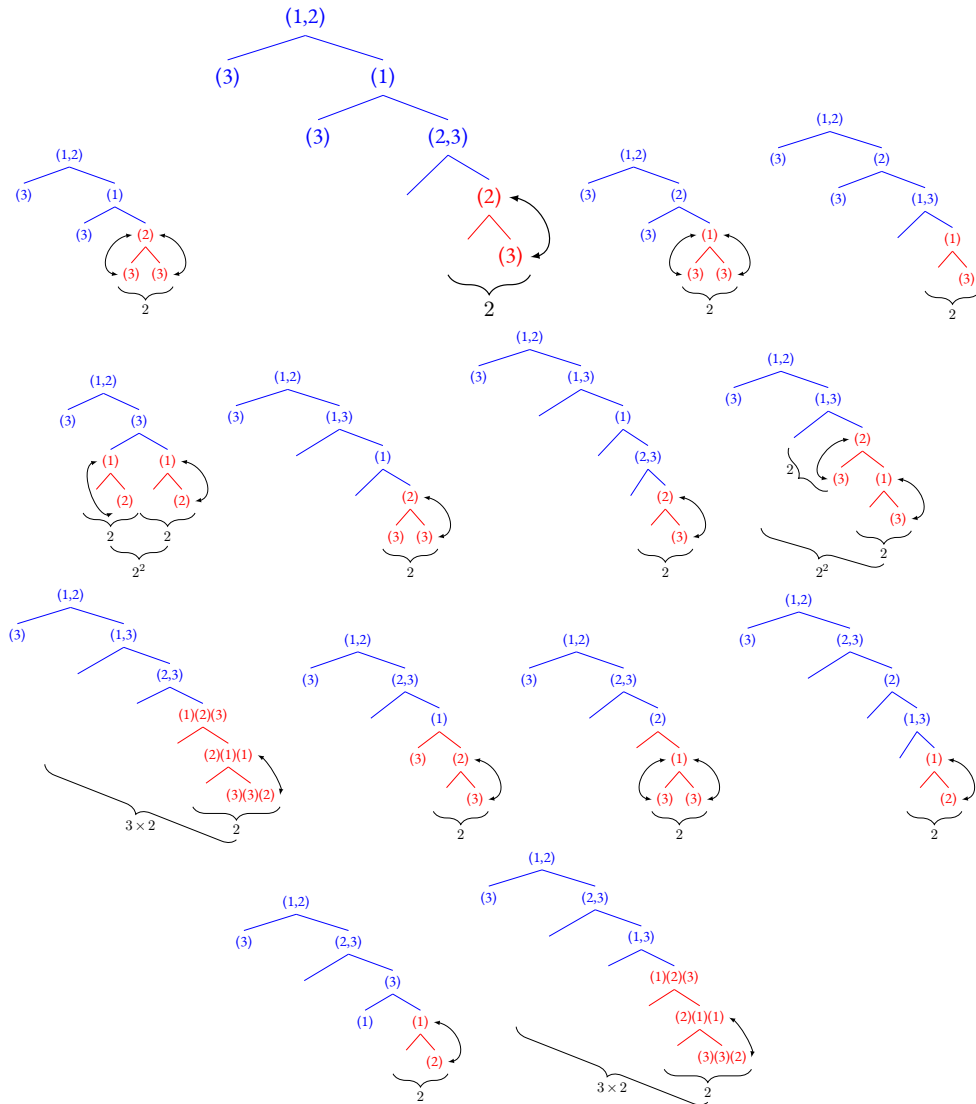


Figure 2.12: Tree representations with a grouping by two elements on the root. For 10 fixed elements, we have 2 possible permutations, for 2 fixed elements, we have 2 possible permutations, and for 2 possible permutations, we have 6 possible permutations. Hence, we finally have $2 \times 10 + 4 \times 2 + 6 \times 2$ for all possible groupings by two elements.

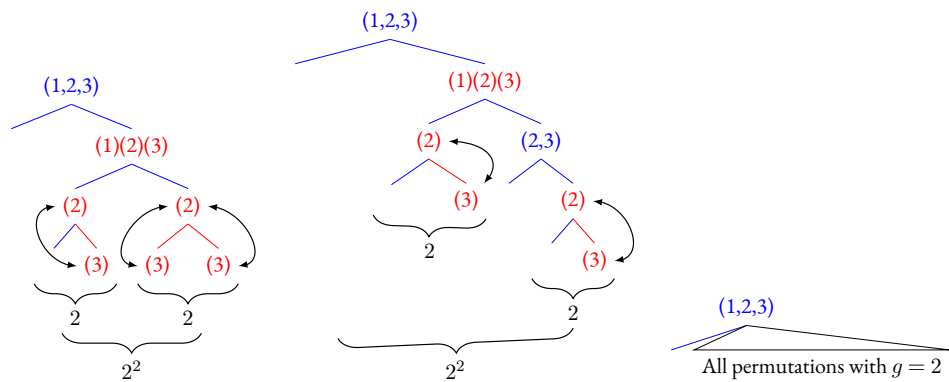


Figure 2.13: Trees representation with a grouping by three elements on the root. For a fixed element at the upper left corner side, we have 2^2 possible permutations. For the upper right corner side, we get 2^2 . We replace the subroot of the fixed trees and get $(2^2 + 2^2) \times 3$. We also have the 40×3 trees from the grouping of two ($g = 2$). Hence, we have $40 \times 3 + (2^2 + 2^2) \times 3$

3 Near-Optimal Pool Testing Under Urgency Constraints

Based on common work with Éric Brier, Megi Dervishi, Rémi Géraud-Stewart and David Naccache.

3.1 Introduction and Motivation

In this section, we discuss pool testing strategies from an information-theoretic perspective. From this point of view, any single test tells us something about the status — positive or negative — of individuals in a population. Since there is no way to learn in this fashion more information than there is, a natural figure of merit for a given testing strategy is to measure what proportion of the total information it collects.

We shall say throughout this section that a strategy is *near-optimal* when this proportion (better described as the ratio of the average number of tests to the entropy of the tested population) exceeds 99%. Note that this threshold is arbitrary and serves to give a concrete instantiation of our methods, yielding relatively simple strategies.

The rest of this section describes testing strategies that are near-optimal in a range of simplified, but realistic situations.

Our aim is to describe these strategies in a way that is immediately applicable to real-world situations, such as detection of SARS-COV-2. Their near-optimality is easy to check, and we introduce a compact graphical notation for them.

Near-optimality does not necessarily imply optimality; however by definition no strategy can outperform ours by more than 1% in terms of number of tests performed.

The mathematical methods and theory that enabled us to design these testing strategies are highly non-trivial, and we defer their complete description to another research.

3.2 Preliminaries

The following subsections will describe a set of testing strategies, called *algorithms* and will provide a high-level summary of the resulting performances. All mathematical computations providing performance estimates are deferred to the appendices of this section.

3.2.1 Graphical Representation of Algorithms

To describe the proposed procedures without ambiguity, we adopt the following graphical representation:

- Each algorithm is represented as a tree read from left to right. Each node has two branches, top and bottom, whose precise meaning is described below.
- Letters at the edges (e.g., A , B , etc.) stand for individuals being pool-tested together at each testing step.
- Leaves indicate samples that are determined *negative* (denoted $-$) or *positive* (denoted $+$). We write $(+, -, -, +, \dots)$ to mean that A is $+$, B and C are $-$, D is $+$ etc.
- A bar over a letter (e.g., \bar{A}) denotes *introduction*, namely the operation consisting in randomly drawing a new individual from the queue and assigning to it the concerned letter (e.g. \bar{A} means: “draw a random individual from the queue and denote it by A ”). See Fig. 2.14.



Figure 2.14: A unary test: draw a random individual from the queue and test it.

- When more than one population is sampled, the use of uppercase and lowercase letters is used to distinguish between the two populations.
- Branches to the bottom represent negative results, and are marked in green. Branches to the top represent positive results, and are marked in red. For instance, Fig. 2.14 shows the classical test of one patient.
- A leaf labeled with the letter **R** means that the concerned individual ought to be “recycled” (or *re-pooled*) in a subsequent test. The reader may be surprised that we re-pool, and may be worried that, in doing so, we somehow lose information. However, this is not the case: as we will detail further below, since in fact, no information was learnt about this patient.
- Finally, edges can carry orange labels (e.g., L4: $\bar{A}, \bar{B}, \bar{C}$). This allows jumping to the concerned edge and repeating a tree branch again. Note that labels are always associated with barred letters (redraw and resume).

Remark 10. The number of individuals being tested (because of introduction), as well as the number of results obtained out of our algorithms (because of re-pooling), depend on successive test results. Thus our algorithms are best interpreted as “streaming” tests that progressively consume an untested population and produce individual test results. As mentioned earlier, “untested” is to be understood in an information-theoretical sense, and is therefore equivalent to stating that we know nothing of its test result. This operation is illustrated in Fig. 2.15.

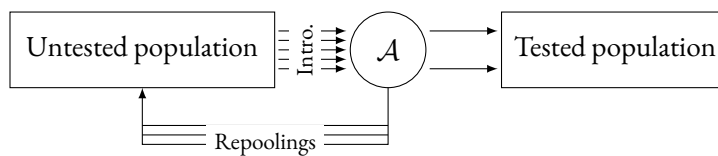


Figure 2.15: A high-level overview of a step during a population test, using one of our algorithms denoted here \mathcal{A} .

Remark 11. It may happen in practice that one or several *introductions* fail, due to the lack of available untested individuals. This can only happen when the remaining untested population is small, which in practical terms means at most a couple of times. When that happens, we can skip the introduction or equivalently we can draw an already-tested, known-to-be-negative individual. The final result is unaffected, as is the total number of tests performed.

3.2.2 Dealing with Urgency Constraints

Unlike other pool-testing strategies where a positive result in a pool yields no information about the individuals consisting the pool, this method allows to guarantee that certain individuals in the pool will get a result. We can, therefore, predict for which position(s) in the testing scheme results are guaranteed. This allows to prioritize

individuals within the testing process without delaying the results of other individuals or adding more load to the system.

For example, in Algorithm \mathcal{A}_3 below, the individuals C and E are guaranteed to receive a testing result (negative or positive), whereas other individuals contribute information to the pool, but themselves may be returned to the group of individuals awaiting the test, and will be tested again with another pool. In Algorithm \mathcal{A}_4 , for example, it is individual D who is guaranteed to get testing results.

3.2.3 Homogeneous and Non-homogeneous Populations

We assume prior knowledge of a risk level, in the form of a probability x that an individual tests positive. Several models can be considered:

- In the *homogeneous population* model, x is the same for every individual;
- In the *non-homogeneous population* model, x depends on the individual being considered (e.g. weight);
- In the *stratified population* model, the population is divided into subgroups, which are assumed to be homogeneous (e.g. age group).

Depending on the model and on the values of x , certain strategies are better than others. At the beginning, we focus on the homogeneous model, providing a set of algorithms that achieve above 99% optimality in a large range of values of x . Then we address the stratified model where we show how to combine the aforementioned algorithms to achieve again at least 99% optimality in a large range of values of x .

3.3 Homogeneous Population Algorithms

The algorithms in this section perform tests in a homogeneous population. We first describe “basic” algorithms, which are then used to generate an infinite family of “compound” algorithms. Finally, we discuss the ranges of probability x over which these algorithms achieve 99% optimality.

3.3.1 Basic Algorithms

3.3.1.1 Algorithm \mathcal{A}_1 .

This algorithm consists in the unary test of a single individual.

3.3.1.2 Algorithm \mathcal{A}_2 .

This algorithm performs a pairwise test with re-pooling, see Fig. 2.16.

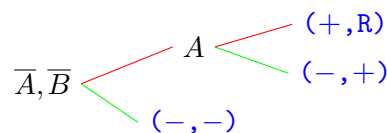


Figure 2.16: Algorithm \mathcal{A}_2 .

3.3.1.3 Algorithm \mathcal{A}_3 .

This algorithm performs an initial three-wise test, with subsequent introductions and re-pooling, see Fig. 2.17.

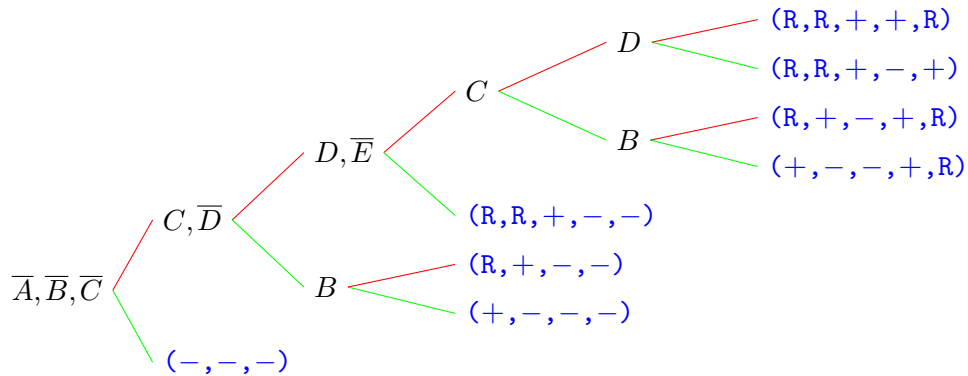


Figure 2.17: Algorithm \mathcal{A}_3 .

3.3.1.4 Algorithm \mathcal{A}_4 .

This algorithm performs an initial four-wise test, then adopts a divide-and-conquer strategy which we generalise below (in Section 3.3.2), see Fig. 2.18. Let us detail the operation of this algorithm. If the first test is negative, we conclude that none of A, B, C and D are infected and proceed with a new set of four subjects. If the first test is positive, we test C and D together. If this second test is positive, we conclude that C, D or both are infected: we test D and conclude as before. If the second test is negative, we conclude that C and D are not infected: we test B and conclude as before.

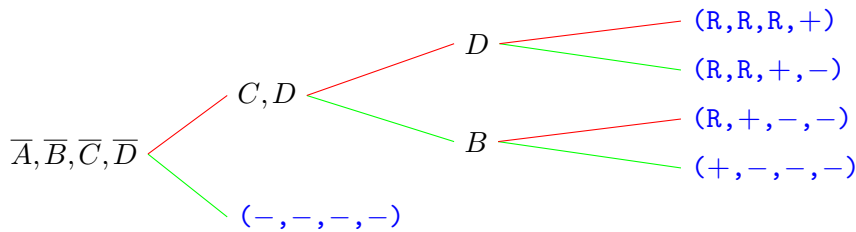


Figure 2.18: Algorithm \mathcal{A}_4 . Note that D is never re-pooled.

3.3.1.5 Algorithm \mathcal{A}_5 .

This algorithm is the most complex of the basic ones, and begins with a five-wise test, see Fig. 2.19.

Let us consider we are testing individuals A, B, C, D and E together.

- If the first test is negative, we conclude that none of the five subjects is infected and can restart the algorithm with a new set of individuals. Otherwise, we test A and B together.
- If the second test is positive, we re-inject C, D and E to the pool of individuals to be tested, test B and conclude as before for A and B . Otherwise, we conclude that at least one subject between C, D and E is infected.

- We pick two new subjects, say F and G , and test E, F and G together.
- If the third test is negative, we conclude that E, F and G are not infected, and that C, D or both are infected and conclude after testing C . Otherwise, we then test C, D and G altogether.
- If this fourth test is negative, we conclude that C, D and G are not infected, implying that E is infected (since CDE was positive) Since E is infected the EFG test brings no information about F , which must be re-injected into the pool of subjects to be tested.
- If the fourth test is positive, we then test G . Otherwise, we conclude that C, D or both are infected (since CDG test was positive) and that E, F or both are infected (since EFG test was positive). We then test D individually and F individually and conclude as before.
- We are left with the case where the test on G alone is positive. While we easily conclude that G is infected, we also conclude that the EFG and CDG tests do not bring any information about C, D, E and F . It remains however the knowledge that CDE had a positive test. We can thus go back to the position we were after the second test, and restart the process with a new individual G' replacing G .

Remark 12. The loopback in the process occurs rarely; the probability that this loop is taken several times is extremely small, however it is not zero. In practice, there may be a limit on the number of times a given individual can be tested. To avoid running in such issues it is possible to abort early by testing C alone, or D and E together.

3.3.2 Compound Algorithms

Using the basic algorithms described in the previous section, we can build new algorithms as follows: choose an algorithm \mathcal{A}_n , and instead of applying algorithm \mathcal{A}_n to individuals, we apply it on samples resulting from *pairs* of individuals. The outputs of \mathcal{A}_n will then need to be re-interpreted: a negative result means both members of the pair are negative, but a positive output for a mix AB means that either A, B or both are infected. As before, we test B . If the test on B is positive, B is infected and we gained no information about A . If test on B is negative, B is not infected but A is infected.

Remark 13. This generic construction yields \mathcal{A}_2 and \mathcal{A}_4 from \mathcal{A}_1 and \mathcal{A}_2 respectively. Therefore, these basic algorithms can be considered redundant.

Starting with the sets of algorithms $\{\mathcal{A}_1, \mathcal{A}_3, \mathcal{A}_5\}$, we get an infinite family of algorithms:

$$\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6, \mathcal{A}_8, \mathcal{A}_{10}, \mathcal{A}_{12}, \mathcal{A}_{16}, \mathcal{A}_{20}, \mathcal{A}_{24}, \mathcal{A}_{32}, \mathcal{A}_{40}, \mathcal{A}_{48}, \mathcal{A}_{64}, \dots$$

3.3.3 Complexity Analysis

Let \mathcal{A}_n be one of the algorithms described above (basic or compound), we are interested in the number $f_n(x)$ which counts how many tests per person are needed on average to get the definitive status (positive or negative) of every individual, as a function of the population risk level x .

3.3.3.1 Algorithm \mathcal{A}_1 . We have, obviously, $f_1(x) = 1$.

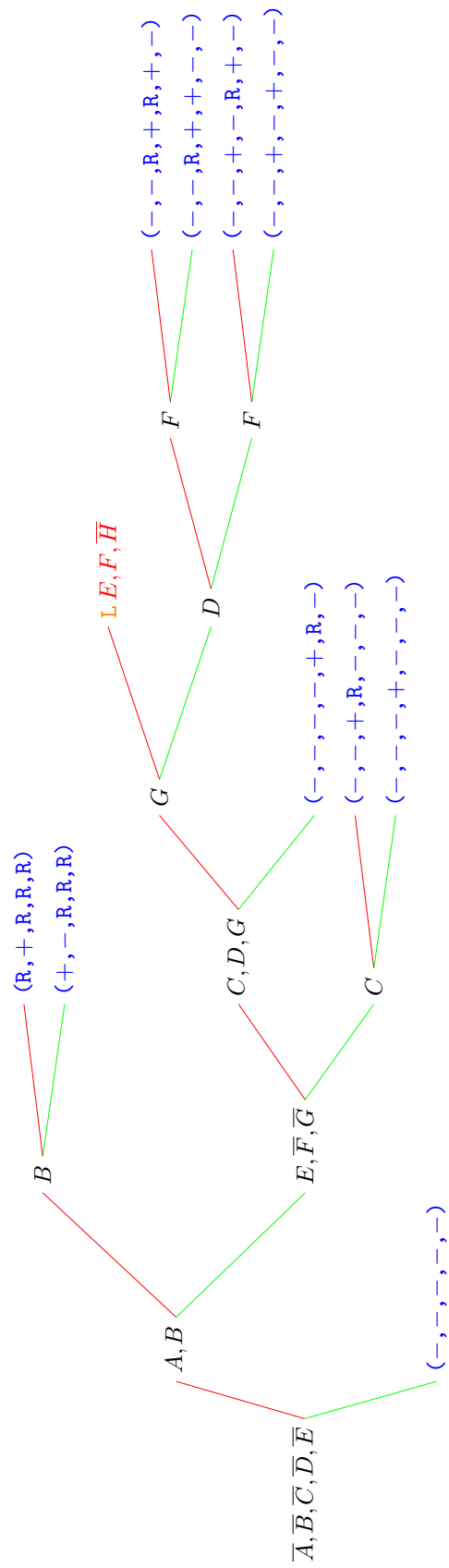


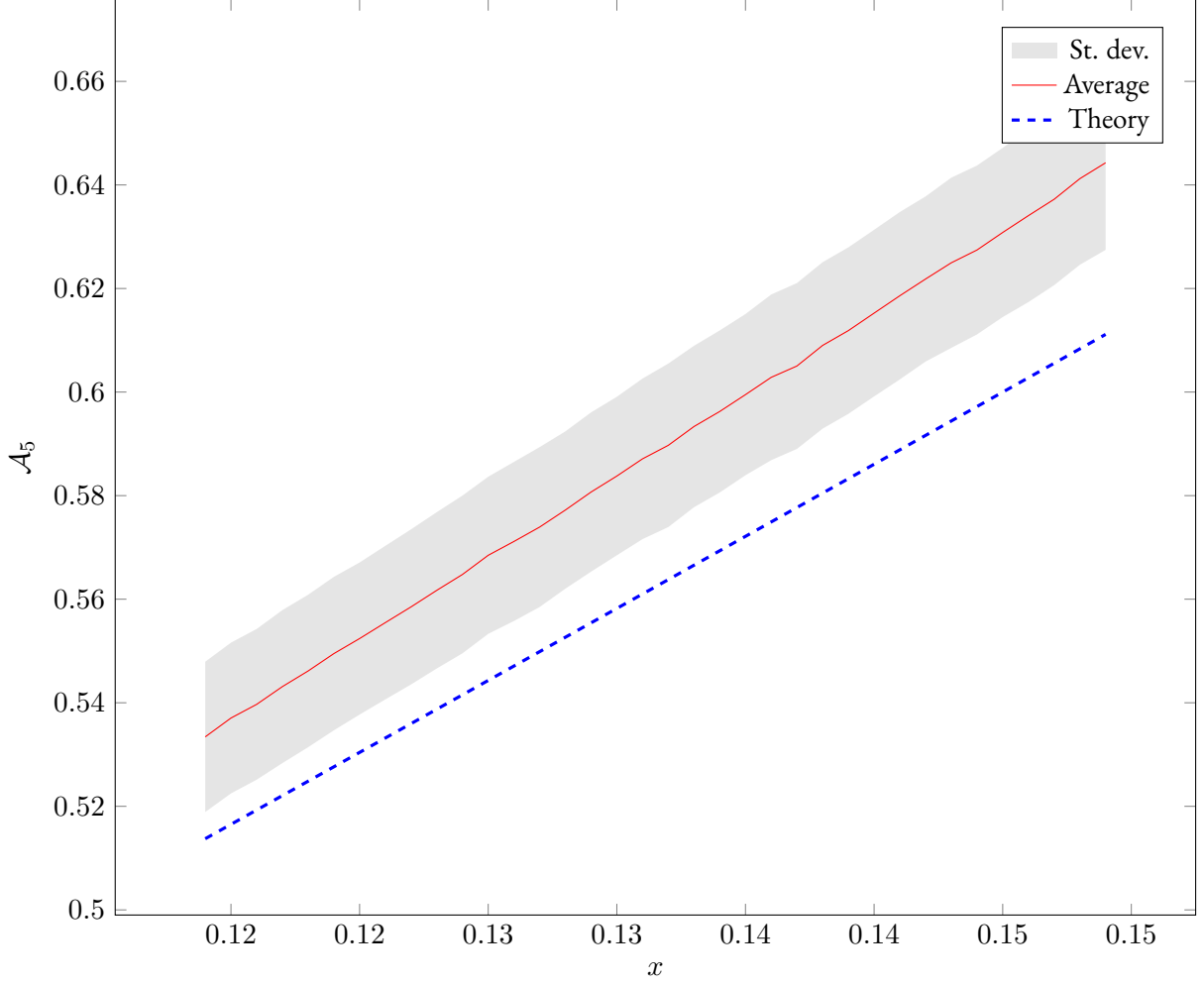
Figure 2.19: Algorithm \mathcal{A}_5 .

3.3.3.2 Algorithm \mathcal{A}_3 . We first compute the probability of each leaf of the graph. For example, the leaf $(-, -, -)$ is reached if, and only if, A , B and C are not infected, which has probability $(1 - \rho)^3$. As a second example, the leaf $(R, R, +, -, -)$ is reached when C is infected, while A and B are not, disregarding the status of A and B . As a result, the probability to reach this leaf of the graph is $\rho(1 - \rho)^2$. Summing the number of tests needed to reach each leaf, weighted by the probability to reach this leaf, gives the average number of test per run of the algorithm. In the same vein, summing the number of known status, weighted by the probability to reach this leaf, gives the average number of patients whose status is discovered, per run of the algorithm. Dividing those two average numbers yields the desired value of average number of tests needed to get the status of one patient. For algorithm \mathcal{A}_3 , the result is:

$$f_3(x) = \frac{2x^4 - 6x^3 + 2x^2 + 6x + 1}{x^3 - 3x^2 + x + 3}.$$

3.3.3.3 Algorithm \mathcal{A}_5 . Analysis is similar to \mathcal{A}_3 , with one additional complication: indeed, there is a possible loop back in the algorithm. A simple way to circumvent this is to expand the loop back and consider an infinite but rather simple graph and proceed as for f_3 . The infinite series that appear have closed loop expressions (and are well known). We end up with:

$$f_5(x) = \frac{3x^6 - 18x^5 + 36x^4 - 24x^3 - 8x^2 + 13x + 1}{(x^2 - x - 1)(x^3 - 5x^2 + 8x - 5)}.$$



3.3.3.4 Algorithms \mathcal{A}_{2n} . For compound algorithms, we can express f_{2n} as a function of f_n . Let us denote x_2 the probability that at least one individual of a pair is infected, which is a simple function of x : $x_2 = 1 - (1 - x)^2 = 2x - x^2$.

The cost of the execution of \mathcal{A}_n on a pair is on average $\alpha = f_n(x_2)$ to get the status of one pair. With probability $(1 - x)^2$, we get the (negative) status of two patients at the cost of α tests on average. With probability $x(1 - x)$, we get the (mixed) status of two patients at the cost of $\alpha + 1$ tests in average. Finally, with probability x , we get the (positive) status of one patient at the cost of α tests in average. The average cost for one run is $x_2 + f_n(x_2)$, while the average number of statuses determined in one run is $2 - x$. We thus have:

$$f_{2n}(x) = \frac{x_2 + f_n(x_2)}{2 - x}.$$

In particular, this allows us to express f_2 and f_4 :

$$f_2(x) = \frac{x^2 - 2x - 1}{x - 2}, \quad f_4(x) = \frac{2x^4 - 8x^3 + 12x^2 - 8x - 1}{(x - 2)(x^2 - 2x + 2)}.$$

3.3.4 Cut-off Points for Basic Algorithms

Using the functions f_n described above, we can identify which algorithm is the best at a given value of x . Because all the f_n are rational functions in x , these regions of dominance are finite unions of intervals — and in this particular case, they are simple intervals. In other terms, we can describe an algorithm's dominance region by specifying a “cutoff value” at which another algorithm becomes superior.

The value γ_1 is the one at which algorithms \mathcal{A}_1 and \mathcal{A}_2 have the same performances, i.e., it is a root of the numerator of the difference $f_1 - f_2$. Therefore:

$$\gamma_1^2 - 3\gamma_1 + 1 = 0$$

Similarly, γ_2 is the cut-off point between \mathcal{A}_2 and \mathcal{A}_3 , thus a root of $f_3(x) - f_2(x)$, which yields the equation:

$$\gamma_2^3 - 4\gamma_2^2 + 5\gamma_2 - 1 = 0$$

Following this approach we obtain equations satisfied by all the cut-off points:

$$\mathcal{A}_2/\mathcal{A}_1 : \gamma_1^2 - 3\gamma_1 + 1 = 0$$

$$\mathcal{A}_3/\mathcal{A}_2 : \gamma_2^3 - 4\gamma_2^2 + 5\gamma_2 - 1 = 0$$

$$\mathcal{A}_4/\mathcal{A}_3 : 2\gamma_3^3 - 7\gamma_3^2 + 7\gamma_3 - 1 = 0$$

$$\mathcal{A}_5/\mathcal{A}_4 : \gamma_4^9 - 10\gamma_4^8 + 42\gamma_4^7 - 96\gamma_4^6 + 127\gamma_4^5 - 91\gamma_4^4 + 21\gamma_4^3 + 14\gamma_4^2 - 9\gamma_4 + 1 = 0$$

$$\mathcal{A}_6/\mathcal{A}_5 : \gamma_5^9 - 10\gamma_5^8 + 44\gamma_5^7 - 112\gamma_5^6 + 179\gamma_5^5 - 178\gamma_5^4 + 98\gamma_5^3 - 16\gamma_5^2 - 8\gamma_5 + 1 = 0$$

which correspond to approximate values:

$$\begin{aligned} \gamma_1 &= 0.381966011250105, & \gamma_2 &= 0.245122333753307, & \gamma_3 &= 0.170516459041503, \\ \gamma_4 &= 0.149636955876700, & \gamma_5 &= 0.113817389150325 \end{aligned}$$

These values are illustrated on Fig. 2.20.

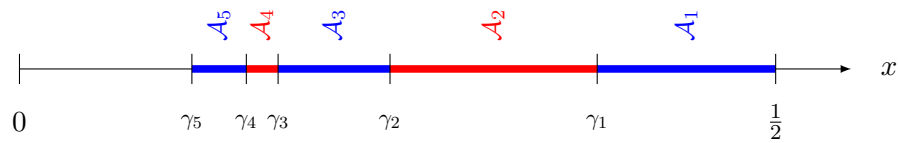


Figure 2.20: Region in which each elementary algorithm reaches > 99% optimality, as a function of probability x .

Fig. 2.20 seems to point two shortcomings of basic algorithms: the region below γ_5 and the region above $x = 1/2$. For the former, we will discuss below how compound algorithms can provide a solution; for the latter, we formulate the following:

Conjecture 1. There is no better homogeneous population algorithm than \mathcal{A}_1 when $x > \gamma_1$: patients are to be tested individually.

Over their respective regions of dominance, we can compute the optimality of each algorithm with respect

to the information-theoretical bound: \mathcal{A}_1 reaches 95.9% for \mathcal{A}_1 , and \mathcal{A}_2 through \mathcal{A}_5 all exceed 99%.

Finally, for every $x < 0.23$, there exists n and $k \in 1, 3, 5$ such that \mathcal{A}_{2^nk} reaches 99% optimality. Unfortunately, this fact does not help in selecting *which* values of n and k to choose for a given value of x .

3.4 Stratified Population Algorithms

The results of the previous sections are very efficient for homogeneous populations with low risk level. However, they may be sub-optimal (by several percents) across some higher risk level ranges.

This section addresses strategies consisting in mixing two groups. When facing more than two groups, the strategies described here can also be used on pairs of groups. Further algorithms can be derived based on the principles described in this section.

Once again, we focus on reaching the (arbitrary) minimal performance of 99%. To avoid unnecessary complexity, we restrict ourselves to consider two populations, with risk levels x and y satisfying $x < y$ and $y < 0.23$. This ensures that we already have at hand a quasi-optimal (i.e., performance above 99%) algorithm for homogeneous populations with risk level y .

We also assume that the low risk population is much larger than the high-risk one. The strategy consists, therefore, of using a mix of subjects to deal with the high-risk ones. Then, we will be left with excess of low-risk patients, that we suggest to deal with as a homogeneous population.

3.4.1 Basic Algorithms

3.4.1.1 Algorithm \mathcal{M}_1 .

This algorithm tests pairs of type Ab with risk level x for A and risk level y for b . Each time such test is negative, one concludes that A and b are not infected. Each time the test Ab is positive, b is sent to a pool of patients with probability $z = y / (x + y - xy)$. This second pool is tested using the best available algorithm for homogeneous population with risk level z . Then, as usual, if b happens to be negative, we conclude that A is positive and when b happens to be positive, we re-pool A .

Fig. 2.21 describes this algorithm, with test b in purple to highlight it is not a direct test on a unique sample.

Let us note $\varphi(z)$ the cost function for best available algorithm for homogeneous population with risk level z . The function φ can be picked amongst the cost functions detailed in Section 3.3.3. The cost for execution of algorithm \mathcal{M}_1 is then

$$1 + (x + y - xy) \cdot \varphi\left(\frac{y}{x + y - xy}\right).$$

One execution of algorithm \mathcal{M}_1 brings surely knowledge about patient b 's status and brings knowledge about patient a with probability $1 - y$. As such, the overall performance of \mathcal{M}_1 is

$$\frac{(1 - y)H(x) + H(y)}{1 + (x + y - xy) \cdot \varphi\left(\frac{y}{x + y - xy}\right)}$$

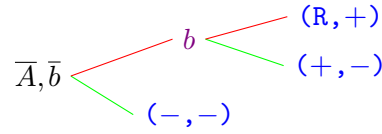


Figure 2.21: Algorithm \mathcal{M}_1 . Mixed pair test with re-pooling

3.4.1.2 Algorithm \mathcal{M}_2 .

The algorithm \mathcal{M}_2 is described in Fig. 2.22.

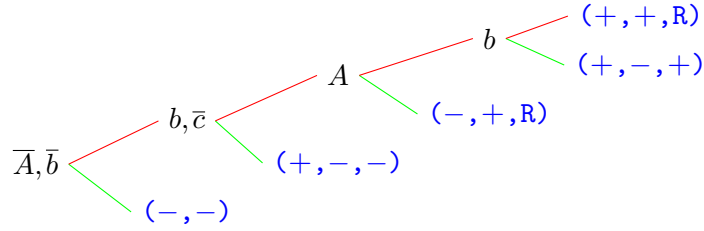


Figure 2.22: Algorithm \mathcal{M}_2 . Mixed pair test with re-pooling

3.4.1.3 Algorithm \mathcal{M}_3 .

We consider a patient A with risk level x and the other patients with risk level y . We will start with testing A and b together. If the result is positive, we will determine the status of b , by testing b with other patients with risk level y . See Fig. 2.23. The cost of running the algorithm is once:

$$\frac{(1+y)(1+x-xy)}{1-y}$$

The algorithm provides knowledge about A with probability $(1-y)$ and the average number of patients with risk level y whose status is determined is:

$$\frac{(1+x-xy)}{1-y}$$

As a result, the performance of the algorithm, in terms of average information obtained per test is

$$\frac{(1-y)^2 H(x) + (1+x-xy) H(y)}{(1+y)(1+x-xy)}$$

3.4.2 Combining Algorithms \mathcal{M}_1 , \mathcal{M}_2 and \mathcal{M}_3

Depending on the values of x and y , one will choose algorithm \mathcal{M}_1 , \mathcal{M}_2 , \mathcal{M}_3 or \mathcal{M}_1 applied to patients with risk level x and pairs of patients with risk level y , followed by an additional test when a pair is found to be positive.

For all risk levels $31.25\% < x < 44.18\%$ and all risk levels $11\% < y < 22\%$, a performance of at least 99% can be reached by one of those four algorithms. For lower risk levels y , the same technique is applied using groups of 2^n patients. There always exist n such that the probability of a group of 2^n has at least one of them infected falls in the range $31.25\% < x < 44.18\%$ and the tests needed to split the 2^n groups in 2^{n-1} have

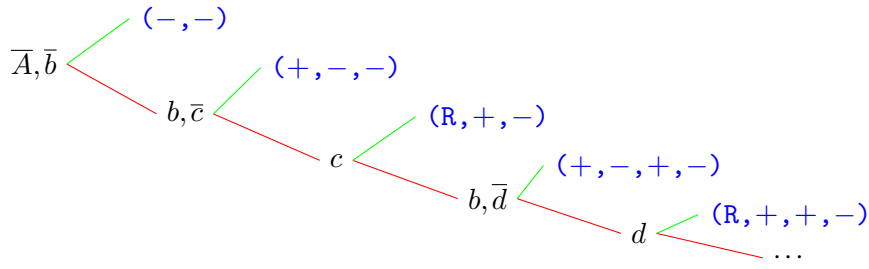


Figure 2.23: Algorithm \mathcal{M}_3 . Mixed population recursive testing

average entropy above 99%.

As a final result, for all risk levels $31.25\% < x < 44.18\%$ and all risk levels $y \leq 22\%$, we have built an algorithm that ensures at least 99% optimality.

3.4.3 Algorithms $\mathcal{M}_{4,n}$

We will now define a family of algorithms that generalizes the algorithm \mathcal{M}_1 . The algorithm $\mathcal{M}_{4,n}$ starts with testing one patient A having risk level x together with n patients b_i each having risk level y . If the test is negative, the $n + 1$ patients are negative, and we are done. In case the test is positive, we test b_n then b_{n-1} then b_{n-2} and so on. Each of these patients is tested using one of the algorithms developed for homogeneous population, using the fact it has an *a posteriori* probability z_i which is a function of x and y . If one of these tests is positive, the patient b_i is positive, patients b_{i+1} to b_n are negative, and we have to re-pool patients A and b_1 to b_{i+1} . The remaining case is that all tests until b_1 are negative. We then conclude that A is positive.

In Section 3.4.2 we addressed the “window” $x \in [0.3125, 0.4418]$, where no known homogeneous algorithm reaches 99% performance. The second “window” where we do not have a quasi-optimal algorithm (99% performance) is the interval $[0.2345, 0.25809]$. In this case, using $\mathcal{M}_{4,2}$, $\mathcal{M}_{4,3}$, $\mathcal{M}_{4,4}$ or $\mathcal{M}_{4,5}$, we can reach 99% performance provided that the risk level y is between 6% and 18% and the associated population is “large enough”. For cases where y is lower than 6%, we use recursively the same trick as before, creating a virtual population of risk level $y' = 2y - y^2$ by considering pairs of patients. When a pair is found positive, we test as usual on element of the pair and conclude as in algorithm \mathcal{M}_2 .

As a final result, for all risk levels $23.45\% < x < 25.809\%$ and all risk levels $y \leq 22\%$, we have built an algorithm that ensures at least 99%.

Combining with Section 3.4.2, as soon as sufficiently large population with risk level y below 18% is available, we can manage a population with any risk level below 50% with efficiency at least 99%.

3.5 Alternative Compound Strategies

This part of the chapter slightly improves performance, at the price of increasing the testing design complexity.

When drawing performance curves, one can notice that algorithm \mathcal{A}_{16} seems inefficient compared to its neighbours, even if it fares better on a range of values for x . Instead of using \mathcal{A}_{16} , we consider \mathcal{A}_{15} :

- It starts with testing groups of 15 subjects. When the first test is positive, we test a subgroup of 6 subjects. We thus end up with a group of 6 or 9 subjects, at least one of which is infected.

- For subgroups of 6, we test the first one using one of the algorithm \mathcal{A}_3 . If negative, we test the second one, again with algorithm \mathcal{A}_3 . Either we identified an infected subject, or we are left with a subgroup of 4 subjects where one of which is infected. We perform two halving steps to conclude.
- For subgroups of 9, we split them in subgroups of 4 or 5 by testing a set of 4 patients. Groups of 5 are again managed by testing one patient with \mathcal{A}_3 . So either we are done, or we are again left with a group of 4 patients and we apply two halving steps.

Over the range of risk level ρ where \mathcal{A}_{16} outperforms \mathcal{A}_{12} and \mathcal{A}_{20} , the new algorithm \mathcal{A}_{15} outperforms \mathcal{A}_{16} . As a result, an improved sequence of algorithms is:

$$\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6, \mathcal{A}_8, \mathcal{A}_{10}, \mathcal{A}_{12}, \mathcal{A}_{15}, \mathcal{A}_{20}, \mathcal{A}_{24}, \mathcal{A}_{30}, \mathcal{A}_{40}, \mathcal{A}_{48}, \mathcal{A}_{60}, \dots$$

Another construction allows for some optimization. The technique is to deal recursively with groups of 5 subjects. When the result is positive, one has to test the first subject of the group of 5, again using recursively previously known algorithm. If this individual is positive, one has to re-pool the other four. If this individual is not infected, we are left with a group of 4 subjects, of which at least one is infected. Two halving steps are performed to conclude.

Still another construction allows for some optimization. The technique is to deal recursively with groups of 9 subjects. When the result is positive, one has to test the first individual of the group of 9, again using recursively the previously known algorithm. If this individual is positive, one has to re-pool the other four. If the individual is not infected, we are left with a group of 8 subjects, one of which is infected. Three halving steps are performed to conclude.

3.6 Open Questions

Beyond the conjectures formulated in the course of this work, there are interesting questions left open for further research:

- We assume perfect knowledge of the population risk x to select the best algorithm. It seems that a slight error in the value of x may in some situations cause us to select one of the neighbouring algorithms. Because the f_n are rational (and hence continuous) this should have a limited effect, however this intuition should be formalised: what is the effect of having an uncertainty on x ?
- Our work builds on the assumption that dilution effects are negligible, and that tests are perfectly accurate. Lifting these hypotheses is left as a question for further research.
- Assume that there are two variants V_1 and V_2 of a disease, and we have tests G (“generic”) and S (“specific”), so that G detects either of the variants and S detects only one. What are the optimal strategies then to correctly identify individuals carrying V_1 and those carrying V_2 ? What if both variants can coexist?

4 Preservation of DNA Privacy During the Large Scale Detection of Covid-19

Based on common work with Marcel Hollenstein, David Naccache, Peter B Rønne, Peter YA Ryan, and Robert Weil.

4.1 Privacy Issues Related to Covid-19

In the attempt to control the virus' spread during the Covid-19 pandemic, mobile software applications (tracing apps) were developed. These apps use digital tracking that monitor contact between individuals, so as to easily identify possible exposure to the virus. Some of these apps are based on tracking the geographical location of app users, thus raising privacy concerns.

The scientific debate concerning privacy of the Covid-19 tracing efforts were intense, especially regarding the choice between centralised and decentralised tracing apps [Vaudenay, 2020], and it has had political implications, such as Germany changing to a decentralised approach [Schwartz, 2020]. Oddly, the privacy concerns regarding Covid-19 testing, however, have not received as much attention even though the privacy at stake is arguably even higher, potentially compromising the privacy of one's DNA.

4.2 The DNA Privacy Problem

In both types of tests (Diagnostic tests and Serological tests), the collected specimen contains the tested person's DNA. DNA is the molecule that carries the genetic instructions of all living organisms. Screening of vast populations for the presence of the virus, inevitably means providing the testing agencies (clinics, governments, airlines, etc.) with sensitive genetic information on a considerable number of individuals. Even if the sample contains a very small amount of DNA, PCR can be used to amplify the DNA and reveal the genome [Kang, 2011]. A USB portable device, the MinION, developed by Zaaijer et al. [Zaaijer, 2017] can accurately identify human cells ("DNA re-identification") by comparing an unknown DNA sample to a collection of known DNA profiles, with 99.9% confidence, within three minutes of DNA sequencing.

DNA samples collected as part of Covid-19 tests are not supposed to be analyzed, and in any case, sensitive medical information is expected to be kept confidential. However, it is common practice to preserve medical samples to be used in further research or for prognosis monitoring.

Often times, tested individuals do not know how these samples will be used in the future. For example, in 2009, it was discovered that Texas had been collecting and storing blood and DNA samples taken from millions of newborns without the parents' knowledge or consent. These samples were used by the state for genetic experiments and for the set up of a database [Waldo, 2010].

DNA databases, or *biobanks*, are being maintained in many countries in the world [Williams, 2018] and they are being used for forensic [Kayser, 2011] and research purposes. An analysis from 2012 indicates that there is no consensus on the the need for consent to use information in biobanks [Master, 2012]. Despite the matter's sensitivity, the information can be accessed. In a research conducted in 2016, 95.7% of 46 biobanks surveyed by [Capocasa, 2016] gave other researchers permission to access their samples. Despite laws and regulations intended to prohibit the re-identification of anonymized data, such as Privacy Rule from HIPAA (Health Insurance Portability and Accountability Act of 1996), the Common Rule from the DHHS (Department of Health and Human Services), and the Human Subject Protection Regulations and 21st Century Cures Act

from the FDA [Lee, 2000], there have been numerous incidents of data breaches (including database hacks and ransomware attacks) in healthcare systems [Pecci, 2017; Monica, 2017; CBC, 2016].

DNA samples stored in databases are usually coded so as to reduce their identifiability. However, it is not impossible to track down the individual “behind” the DNA. For example, Malin and Sweeney [Malin, 2000; Malin, 2001] demonstrated that even if kept confidential, and without being linked to identifying personal or demographic details, DNA information can be traced to the tested patients using inferences drawn from the DNA information.

The fact that stored DNA can be linked to the person is especially problematic given the possible use there could be for this information. Knowing a person’s DNA provides information about racial features, potential diseases or life span expectancy. This information can attribute to genetic discrimination. For example, an employer may refuse to hire someone based on the likelihood that they will become ill. Similarly, exposing one’s genetic information may impact their eligibility to life or health insurance, or to increase their premia [Board, 1998]. In this sense, the potential transfer of sensitive genetic information to a third party raises significant ethical and legal issues [Cambon-Thomsen, 2004]. Moreover, the collection of DNA into databases can “raise human rights concerns, including potential misuse of government surveillance (for example, identification of relatives and non-paternity) and the risk of miscarriages of justice” [Cannataci, 2016].

Beyond the discrimination against individuals based on their genetic profiles, human rights activists have been protesting against the mass collection of DNA samples from citizens by governments. In the past years, China has been collecting DNA samples from citizens as part of mandatory medical examinations [Feng, 2017]. Human Rights Watch activists are worried about the use of this information for “surveillance of persons because of ethnicity, religion, opinion or other protected exercise of rights like free speech” [Haas, 2017].

The gathering of DNA information by countries has raised concern regarding the way this information can be used to impact populations based on genetic characteristics. Moreover, this information may be deployed not only domestically, but internationally [Mosher, 2019a]. DNA information can be used to attack strategically identified persons, such as diplomats, politicians, high-ranking federal officials, or military leadership, or even to bio-engineer a disease that would be fatal to some races but not to others [Mosher, 2019b].

Refusal to undergo medical tests or procedures because of the fear of exposing genetic information to hostile entities may hinder medical and scientific attempts to treat diseases and learn about them for the sake of mankind. During the pandemic, for example, it has been reported that Israel had revoked a deal with a company selling Covid-19 testing equipment out of concern about granting the company access to Israelis’ genetic information [Schulman, 2020]. Especially during the pandemic, when Covid-19 tests were prevalent and even became mandatory in different settings (e.g. prior to international flights) [Cripps, 2020], the need to find a way to conduct these tests without compromising our genetic information’s safety arises. The fact that test samples are often sent to be analyzed in another country, e.g. [BBC, 2020], reinforces the need to form a testing method that ensures that the samples sent do not contain identifiable DNA traces.

The rest of the section is organized as follows: in the next subsection, we describe a theoretical safety model inspired by the cryptographic notion of *Indistinguishability under Chosen Plaintext Attack*. In 4.3, we describe a testing scheme applying the theoretical safety model to create privacy preserving medical tests, and elaborate on different approaches that could be applied. Finally, subsection 4.4 concludes the topic.

4.3 Suggested Solution - Theoretical Model

Before describing the solution, let us provide the intuition behind our idea.

We aim to develop a DNA privacy-preserving test, or in other words, a test method that would yield the same results, but that the DNA in the specimen tested, or in any residue of the specimen collected, will be undetectable. We hence wish to develop a test procedure \mathcal{T} such that:

$$\mathcal{T}(\text{DNA} \# V) = \{\text{positive}, \text{res}_{\text{positive}}(\text{DNA}, V)\} \text{ and } \mathcal{T}(\text{DNA}) = \{\text{negative}, \text{res}_{\text{negative}}(\text{DNA})\}$$

The positive and negative denote the virus' presence or absence, respectively (the test's result). $\#$ denotes mixing substances¹¹. res denotes the test's residue, i.e. whatever is left after the test procedure has been completed. While some protocols may require this residue to be treated as bio-hazardous waste and be destroyed, residues are often preserved for future research or other purposes. Therefore, res is where unwanted DNA may be present, and is our main concern.

In Covid-19 tests, the sequence of operations is independent of the virus' presence. However, in other types of tests, the virus' presence may influence the sequence of operations done during the test, hence in all generality we distinguish two types of residues in our model. In any case, the residue will differ by the remains of the virus (or lack thereof).

The attacker's definition (\mathcal{A}) is inspired by the cryptographic notion of Indistinguishability under chosen-plaintext attack (IND-CPA):

\mathcal{A} selects two DNA samples DNA_0 and DNA_1 and submits them to a challenger \mathcal{C} . \mathcal{C} picks a random $b \in \{0, 1\}$ and manufactures the samples:

$$\sigma_{b,\text{positive}} = \text{DNA}_b \# V \text{ and } \sigma_{b,\text{negative}} = \text{DNA}_b$$

\mathcal{C} runs \mathcal{T} on $\sigma_{b,\text{positive}}, \sigma_{b,\text{negative}}$ and gets:

$$\mathcal{T}(\sigma_{b,\text{positive}}) = \{\text{positive}, \text{res}_{\text{positive}}\} \text{ and } \mathcal{T}(\sigma_{b,\text{negative}}) = \{\text{negative}, \text{res}_{\text{negative}}\}$$

\mathcal{A} gets $\{\text{positive}, \text{res}_{\text{positive}}\}, \{\text{negative}, \text{res}_{\text{negative}}\}$, performs any state of the art analyses and outputs a guess b' .

\mathcal{A} 's advantage is defined as:

$$\text{Adv} = 2|\Pr[b = b'] - \frac{1}{2}|$$

When \mathcal{A} has no advantage in learning DNA information, his only strategy is to guess b' at random. In that case, $\Pr[b' = b] = \frac{1}{2}$ and hence $\text{Adv} = 0$.

When \mathcal{A} always correctly determines b' , we have $\Pr[b' = b] = 1$, i.e. $\text{Adv} = 1$.

Note that when \mathcal{A} is always wrong, we have $\Pr[b' = b] = 0$ and hence $\text{Adv} = 1$. Such an \mathcal{A} is effectively as powerful as an \mathcal{A} who always finds the correct answer as it suffices to negate his response to get a perfect adversary.

¹¹e.g. water $\#$ CO₂ = soda

In other words, $0 \leq Adv \leq 1$. The higher the advantage, the more powerful \mathcal{A} is. We define \mathcal{T} as IND-C.DNA.A¹² secure¹³ if $Adv < 10^{-3}$.

4.4 How to Build IND-C.DNA.A Tests?

We assume that the test procedure needs to be built in such a way that the tested person can trust that once the specimen is collected, the DNA will not be identifiable. Therefore, we suggest to add a testing procedure, step \mathcal{T}_0 , which will be performed in the patient's presence. We suggest two approaches in which this step can be done: Mixing and Destroying.

4.4.1 DNA Mixtures

We suggest to use a testing kit containing a mixture m of DNA samples, thus making it more difficult to analyze, or *profile*.

The complexity of a DNA mixture is determined by the number of people who contributed DNA to the mixture, the amount of DNA that each of them contributed, and the level of DNA degradation. More contributors make a mixture more complex, and therefore, more difficult to interpret [Press, 2019]. DNA profiling requires the comparison of short segments of DNA, called *alleles*, which vary from person to person. As part of the DNA profiling process, the DNA is amplified and the alleles are represented on a graph showing *peaks*. The positions of those peaks indicate which alleles are present, and thus the graph is a visual representation of the DNA in question. The DNA profiling task is based on the comparison of the pattern of those peaks. Small amounts of DNA derived from various contributors add “noise”, called *drop-in*, which makes the comparison process more complicated. The greater the number of contributors is, the more complicated the task of identifying which peaks go with which contributor. In addition, the PCR process during copying reaction by the DNA polymerase creates small peaks, called *stutter products*, which are sometimes the same lengths as PCR products. This can make the determination of whether a small peak is a real peak from a minor contributor or a stutter products even more difficult.

We propose, therefore, to increase this complexity by adding a DNA mixture to the specimen collected. Any analysis performed will be done on the mixture, and not on the individual DNA sample, thus making it more difficult to profile the DNA.

There are four possible scenarii in attempting to identify the DNA in the mixture:

1. The DNA of the victim x is known, and the composition of the mixture m is also known. The challenge is to determine if x is in the mixture $m \# x$ or not.
2. The DNA of the victim x is known, but the composition of the mixture m is unknown. The challenge is to determine if x is in the mixture $m \# x$ or not.
3. The DNA of the victim x is unknown, but the composition of the mixture m is known. The challenge is to isolate the DNA of the victim, x .

¹²Indistinguishable under Chosen DNA Attack.

¹³The limit can be changed according to the test's acceptable level.

scenario	victim DNA x	hiding mixture m	attacker's goal
(A)	known	known	confirm x
(B)	known	unknown	confirm x
(C)	unknown	known	learn x
(D)	unknown	unknown	learn ¹⁴ x

Table 2.3: Attack and protection scenarii.

4. The DNA of the victim x is unknown, and the composition of the mixture m is also unknown. The challenge is to profile (separate) all the $\|m\| + 1$ DNAs in the mixture $m \# x$ so as to learn the DNA of the victim x with probability $\frac{1}{\|m\|+1}$.

As is the case in cryptology, the above scenarii could be generalized and refined. For instance, one may consider a scenario where \mathcal{A} is allowed to perform v (potentially adaptive) experiments with different mixtures m_0, \dots, m_{v-1} and an identical target DNA x etc. Whilst interesting in theory, we did not consider such extensions very relevant to “real world” settings.

Scenarii (A) and (B) represent a situation where the DNA of the individual is already known, and the challenge is to authenticate its presence in the mixture. Authentication methods are commonly used in the forensics field, where DNA found in a crime scene is compared to that of a suspect. For example, Homer et. al [Homer, 2008] have demonstrated that it is possible to identify the presence of genomic DNA of specific individuals within a series of highly complex genomic mixtures, including mixtures where an individual contributes less than 0.1% of the total genomic DNA.

In this section, we address the option of attempting to identify the DNA of the individuals in the mixture when they are unknown to the attacker (scenarii (C) and (D)). Identification methods are intended to reveal the identity of one contributor in a mixture. Currently, these methods are achieved by comparing DNA samples to known profiles in a database. We propose solutions to prevent the possibility of identifying the genetic profile of an individual by an attacker.

Before we proceed, we would like to introduce a subtle distinction between the equality relationship ($=$) in mathematics and the chemical relationship \simeq consisting in comparing two molecular mixtures.

We denote by $a = b$ an exact equality between the chemical components a and b . However, $a \simeq b$ will denote the fact that a cannot be distinguished from b using current laboratory equipment with very high probability (e.g. 99%).

4.4.1.1 Dilution (scenario (C)):

The idea behind this technique is to add the sample into a pre-prepared mixture containing other samples or other DNAs, thus making DNA less identifiable: mislead \mathcal{A} by reducing his advantage, exploiting the difference between $=$ and \simeq . The most plausible way to do so consists in adding to \mathcal{T} , a fixed mixture of k (e.g. $k = 20$) human DNAs taken from existing DNA samples, or animal DNA. Adding the DNA sample to a fixed mixture of DNA would make the process of identification significantly more complicated. However, using a fixed mixture of DNA grants a few possible advantages to \mathcal{A} . First, if the composition of the mixture is known

¹⁴with probability $\frac{1}{\|m\|+1}$.

to \mathcal{A} , identification of the added sample would be a relatively simple task. Second, even without being familiar with the mixture's composition, the characteristics of the contributors need to be taken into consideration to avoid easy identification. For example, race factors can influence the ease with which a sample can be identified. Thus, using a fixed DNA mixture will make the distinguishing of the DNA of the victim x more complicated, but not impossible.

4.4.1.2 Randomizing (scenario $\textcircled{\text{D}}$):

Another workaround, frequently used in cryptology, consists in adding randomness to \mathcal{T} . The idea behind randomizing is using a random mixture of DNA into which the sample is added. By doing so, any fixed DNA defines the distributions of residues $D_{\text{positive,DNA}} = \{\text{res}_{\text{positive}}(\text{DNA}, V)\}$ and $D_{\text{negative,DNA}} = \{\text{res}_{\text{negative}}(\text{DNA})\}$ obtained by testing this specific fixed DNA over and over again using \mathcal{T} .

We design \mathcal{T} in such a way that $\forall \text{DNA}$, the following distributions are indistinguishable:

$$\text{res}_{\text{positive}}(\text{DNA}, V) \sim \text{res}_{\text{positive}} \quad \text{and} \quad \text{res}_{\text{negative}}(\text{DNA}) \sim \text{res}_{\text{negative}}$$

This mixture is not known to \mathcal{A} , and even the number of contributors comprising the mixtures varies. This prevents \mathcal{A} from learning through repeated experimentation. Randomizing makes the profiling task more complicated, because \mathcal{A} will have no prior knowledge about the DNA characteristics. In DNA profiling, this is referred to as lack of *Framework of Circumstances*, [Forensic-Science-Regulator, 2018], which is one of the factors that hinder DNA profiling.

The model here assumes that when m is manufactured, each individual test kit is randomized (e.g. by the addition of a different random assortment of DNA material) so that different tests of the same patient will yield residues that do not leak information about the patient's DNA.

Current profiling methods in use in forensics allow analyzing a mixture of DNA and determining the number of DNA samples mixed into the mixture. However, separating an individual DNA from a mixture of an unknown number of contributors of unknown DNA profiles is a much more complicated task. In order to profile complex DNA mixtures, a software is used for computing the probability distribution for the number of contributors [Taylor, 2014]. If the number of contributors is unknown, the computational load will be considerably higher. Therefore, the Randomizing method could be applied to mask DNA and achieve our goal.

Another way of masking the DNA in question by adding randomization to the solution is to add an *allelic ladder* directly to the sample solution. An allelic ladder is an artificial mixture of the common alleles present in the human population, and it is commonly used to identify alleles in genetic profiles by comparison with peaks.

4.4.2 Destruction (all scenarii):

Another way to ensure that an attacker cannot access the DNA is to destroy it, thus making it unidentifiable. We start by observing that IND-C.DNA.A cannot exist if $\exists \text{DNA}_0, \text{DNA}_1$ such that:

$$\text{res}_{\text{positive}}(\text{DNA}_0, V) \not\sim \text{res}_{\text{positive}}(\text{DNA}_1, V) \quad \text{or} \quad \text{res}_{\text{negative}}(\text{DNA}_0) \not\sim \text{res}_{\text{negative}}(\text{DNA}_1)$$

Simply because \mathcal{A} can run the test by himself and compare the resulting residue to the challenger's residue, it follows that \mathcal{T}_0 must destroy human DNA while allowing subsequent testing of V .

To ensure that no DNA traces remain in the sample we suggest to destroy the DNA, leaving the RNA

intact for the test. We propose, therefore, a model in which all human DNA is destroyed before the rt-PCR amplification. This can be done by treating the samples with DNase, an enzyme that selectively degrades DNA [Sato, 2014]. DNase eliminates DNA from RNA preparations prior to sensitive applications, such as rt-PCR. Within 10 to 20 minutes [Huang, 1996; Biolabs, 2020] there will be no identifiable DNA. To ensure that viral RNA remains intact, the DNase then needs to be inactivated by inclusion of removal reagents [Ambion,] or by using heat inactivation [Wiame, 2000].

Following this process, we suggest a verification of the DNase's effectiveness. This verification process can be done by inducing reaction causing a colour change which could be visible by the patient. One way of doing so could be by applying gold nano-particles (AuNPs) interconnected by DNA duplexes. Without DNase activity, the AuNPs tend to cluster, displaying a blue colour. When DNase is present, it cleaves these duplexes, causing them to spread, which leads to a colour change from blue to red [He, 2017; Baptista, 2008; Xu, 2007]. Another method that could be applied is using a fluorophore-quencher system [Su, 2013]. In the absence of DNase, the fluorophore is in close proximity of the quencher and hence no fluorescence is visible. In the presence of DNase, DNA is hydrolyzed and the fluorophore is free to circulate in the solution, creating a fluorescent reaction which can be easily monitored and detected as an output signal. The patient can witness the change in colour and be convinced that all DNA has been removed. Once this step is complete, the patient is free to go, and the sample is ready for testing.

Before we conclude, we have to consider the case of mutually distrusting parties. In this scenario, the tester wants to ensure that the test provides accurate information (i.e. that the virus will be detected, if present), while the patient may not trust the tester or the test kit. Theoretically speaking, we could have the patients provide their own DNase. Putting aside the practical logistic difficulties and unlikelihood of this solution, this solution poses a risk of a patient intentionally using a chemical killing both DNA and virus (for example, in a scenario of being virus-free as a condition to board a flight or enter a country). On the other hand, if the DNase is provided by the testing agency, the patient may suspect that another chemical is used that simply emulates the colour change. To solve this dilemma, we can have the patient be sampled twice using a classical "cut-&-choose" approach. To both samples the DNase and supplementary chemicals are added. The tested person then chooses one of the samples randomly, and adds his own chemical to verify the presence of DNase. The other sample is then used for testing. This allows the tested person to detect a maliciously generated test kit with probability $\frac{1}{2}$, and of course these odds can be improved, but at the cost of multiplying the number of samples used.

To ensure the integrity of the process, a positive process control method could be integrated. This could be done, for example, by identifying, at the end of the process, a specific reagent or a known human target which will be deposited at the beginning of the analytical process. The presence of the target at the end of the process will allow concluding that a negative result obtained is stemming from absence of the virus, and not from a malfunction in the reaction or process.

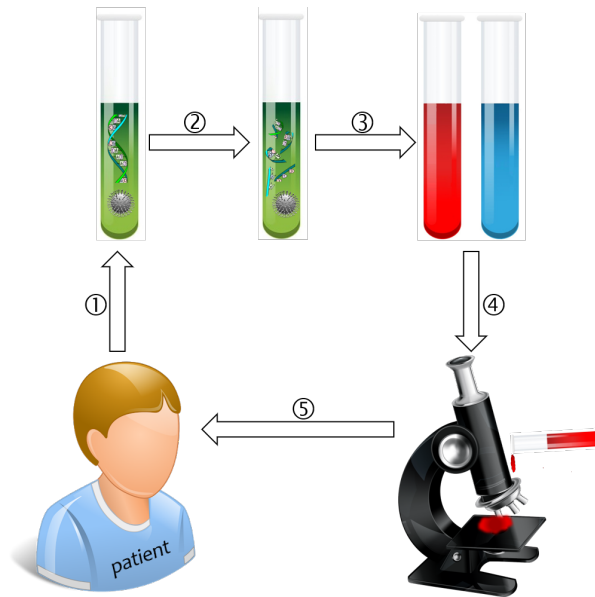


Figure 2.24: Testing following DNA destruction: ① Sample is taken from the patient using swab \Rightarrow ② DNase is applied to the sample, destroying DNA \Rightarrow ③ A colourimetric method is applied \Rightarrow ④ This demonstrates that no DNA traces are present (Blue indicates DNA traces, red means that DNA was properly destroyed) \Rightarrow ⑤ The patient is now convinced that the sample can be analyzed without risk of exposing his DNA.

Note that we could also audit test kits in general, but this relies on trusted third parties, or a public procedure.

4.5 Discussion

This section described a methodology in which biological specimens containing DNA taken from patients can be processed while securing the safety and confidentiality of the DNA information contained in the specimen. Our model relieves the patient of the expectation to *trust* the testing entity.

While the medical system is still based on the patients' confidence and trust in the clinician, in the past years there has been a shift towards more informed patients expecting to have more involvement and control over processes and decision-making [Rowe, 2006].

Mass Covid-19 testing performed all over the world nowadays highlights the need for more secure types of tests. The model proposed in this section can be applied not only to Covid-19, but to other types of tests where DNA is extracted but not necessary to obtain test results. As DNA is present in any specimen collected from the human body, every lab test has the potential of exposing one's sensitive genetic information. The idea described in this section will need to be adapted so as to provide protection to other types of tests.

While public awareness regarding the need to protect genetic information grows, the ability to perform successful profiling using smaller amounts of DNA increases. The recent developments in the field of DNA profiling now allows to analyze even minute amounts of DNA, called *trace DNA* or *touch DNA*. Small amounts of DNA can be found on any surface; people shed DNA on any object or surface they touch. In this sense, one may claim that protecting DNA information is impossible. However, it is important to note the difference between analyzing *trace amounts* of DNA, and analyzing the content of a test tube containing body fluids or mucosa. Let us consider a hypothetical case in which an attacker is interested in the DNA of a specific person. An attacker could try to retrieve trace DNA from objects touched by that person, a cup the person drank from

etc., but the amount of DNA retrieved would be much smaller, and the process of analyzing this DNA would be significantly more difficult. In addition, it is important to note that since, in the case of medical tests, the specimen is sent for analysis in a lab, the likelihood of the DNA to be profiled increases, thus increasing the risk.

At this stage, we only offer a theoretical blueprint. Future work will include laboratory experiments demonstrating that the validity of the test is not negatively impacted by the added security phase \mathcal{T}_0 (i.e. DNA mixture or DNA destruction).

Chapter 3

Experimental Mathematics

1 A Note on the Ramanujan Machine

Based on common work with Éric Brier and David Naccache.

1.1 Introduction

The Ramanujan Machine project [Raayoni, 2021; David, 2021; Raayoni, 2019] detects new expressions related to constants of interests, such as ζ function values, γ and various algebraic numbers (to name a few).

In particular the project lists several of conjectures¹ concerning values of the ζ function.

We show that many of the relations detected by the Ramanujan Machine Project stem from a specific algebraic observation and show how to generate and machine-prove infinitely many.

This provides an automated proof and/or an explanation of many of the relations listed as conjectures by the project (although not all of them).

1.2 Theoretical Preamble

Consider continued fractions defined by the formulae

$$\begin{cases} p_n = b_n p_{n-1} + a_n p_{n-2} \\ q_n = b_n q_{n-1} + a_n q_{n-2} \end{cases} \text{ with the initial conditions: } \begin{cases} p_{-1} = 1 \\ p_0 = b_0 \end{cases} \text{ and } \begin{cases} q_{-1} = 0 \\ q_0 = 1 \end{cases}$$

Let f and g be two functions from which we build $\forall n \geq 1$:

$$\begin{cases} a_n = -f(n)^2 \\ b_n = \frac{f(n+1)g(n+1) + f(n)g(n-1)}{g(n)} \end{cases} \text{ with the initial condition: } b_0 = f(1)g(1)$$

We further require f and g to be nonzero for positive integers.

¹http://www.ramanujanmachine.com/wp-content/uploads/2022/07/results_different_zeta_orders.pdf

Define the sequence \mathbf{F}_n by:

$$F_n = \prod_{i=1}^{n+1} f(i) \text{ with the initial conditions: } \mathbf{F}_{-1} = 1 \text{ and } \mathbf{F}_0 = f(1)$$

We will now prove by induction that

$$\forall n \geq -1, p_n = \mathbf{F}_n g(n+1)$$

The above initial conditions ensure this equality for $n = -1$ and $n = 0$. Assume that the hypothesis holds true for all values below n and let us compute:

$$\begin{aligned} p_n &= b_n p_{n-1} + a_n p_{n-2} \\ &= b_n \mathbf{F}_{n-1} g(n) - f(n)^2 \mathbf{F}_{n-2} g(n-1) \\ &= \mathbf{F}_{n-1} (f(n+1)g(n+1) + f(n)g(n-1)) - f(n) \mathbf{F}_{n-1} g(n-1) \\ &= \mathbf{F}_{n-1} f(n+1)g(n+1) \\ &= \mathbf{F}_n g(n+1) \end{aligned}$$

as desired, thereby proving a closed form for the convergents p_n .

We will now handle q_n by first remarking that:

$$\frac{q_n}{p_n} - \frac{q_{n-1}}{p_{n-1}} = \frac{q_n p_{n-1} - q_{n-1} p_n}{p_n p_{n-1}}$$

The recurrence conditions for p_n and q_n ensure that:

$$q_n p_{n-1} - q_{n-1} p_n = -a_n (q_{n-1} p_{n-2} - q_{n-2} p_{n-1})$$

Induction yields:

$$q_n p_{n-1} - q_{n-1} p_n = \left(\prod_{i=1}^n -a_i \right) (q_{-1} p_{-2} - q_{-2} p_{-1})$$

The last term is equal to 1 given the initial conditions. The n -term product is, by the definition of the sequence a_n , equal to $(\mathbf{F}_{n-1})^2$.

We hence get:

$$\begin{aligned} \frac{q_n}{p_n} - \frac{q_{n-1}}{p_{n-1}} &= \frac{(\mathbf{F}_{n-1})^2}{p_n p_{n-1}} \\ &= \frac{(\mathbf{F}_{n-1})^2}{\mathbf{F}_n g(n+1) \mathbf{F}_{n-1} g(n)} \\ &= \frac{1}{f(n+1)g(n)g(n+1)} \end{aligned}$$

Once again, by induction (taking into account the initial values), we get:

$$\frac{q_n}{p_n} = \sum_{i=0}^n \frac{1}{f(i+1)g(i)g(i+1)}$$

The limit L of the continued fraction is thus given by the equation:

$$\frac{1}{L} = \sum_{n=0}^{\infty} \frac{1}{f(n+1)g(n)g(n+1)}$$

It suffices now to resort to standard partial fraction decomposition to get relations such as those given by the Ramanujan Machine Project. This is done automatically by symbolic computation software such as Mathematica to avoid tedious yet standard formula manipulation by hand.

Example 1.1: The Ramanujan Machine identity $\zeta(4) + 4\zeta(3) - 8$

Consider the Ramanujan Project identity^a where $a_n = -n^8$ and $b_n = n^4 + (n+1)^4 + 2(n^2 + (n+1)^2)$. Posing $g(n) = \eta_1 n + \eta_0$ and identifying we get:

$$b_n g(n) - ((n+1)^4 g(n+1) + n^4 g(n-1)) = (1 + 2n + 2n^2)(2\eta_0 - \eta_1) = 0$$

Which gives the solution $\{\eta_1, \eta_0\} = \{2, 1\}$ (the first example processed by our code).

^aNote that our notations of a_n and b_n are reversed with respect to theirs

The last step connecting the observation and the Ramanujan Machine Project is somewhat technical we hence break it down into successive steps:

1.2.0.1 Step 1: The Ramanujan Machine Project considers that the a_n, b_n defining the continued fraction are polynomials. Because nothing in our analysis imposes that the a_n, b_n are polynomials we can write:

$$\begin{aligned} \frac{1}{L} &= \frac{f(1)g(1) + f(0)g(-1)}{g(0)} - \frac{f(1)^2}{\frac{f(2)g(2) + f(1)g(0)}{g(1)} - \frac{f(2)^2}{\frac{f(3)g(3) + f(2)g(1)}{g(2)} - \frac{f(3)^2}{\frac{f(4)g(4) + f(3)g(2)}{g(3)} - \frac{f(4)^2}{\frac{f(5)g(5) + f(4)g(3)}{g(4)} - \dots}}} \\ \frac{1}{L} &= \frac{f(1)g(1) + f(0)g(-1)}{g(0)} - \frac{g(1)f(1)^2}{\frac{f(2)g(2) + f(1)g(0)}{1} - \frac{g(2)g(1)f(2)^2}{\frac{f(3)g(3) + f(2)g(1)}{1} - \frac{g(3)g(2)f(3)^2}{\frac{f(4)g(4) + f(3)g(2)}{1} - \frac{g(4)g(3)f(4)^2}{\frac{f(5)g(5) + f(4)g(3)}{1} - \dots}} \end{aligned}$$

$$\frac{g(0)}{L} = \frac{f(1)g(1) + f(0)g(-1)}{1} - \frac{g(1)g(0)f(1)^2}{\frac{f(2)g(2) + f(1)g(0)}{1} - \frac{g(2)g(1)f(2)^2}{\frac{f(3)g(3) + f(2)g(1)}{1} - \frac{g(3)g(2)f(3)^2}{\frac{f(4)g(4) + f(3)g(2)}{1} - \frac{g(4)g(3)f(4)^2}{\frac{f(5)g(5) + f(4)g(3)}{1} - \dots}}$$

$$\frac{g(0)}{L} = f(1)g(1) + f(0)g(-1) - \frac{g(1)g(0)f(1)^2}{f(2)g(2) + f(1)g(0) - \frac{g(2)g(1)f(2)^2}{f(3)g(3) + f(2)g(1) - \frac{g(3)g(2)f(3)^2}{f(4)g(4) + f(3)g(2) - \dots}}$$

1.2.0.2 Step 2: In the Ramanujan Project many formulae have an a_n (in our notations, i.e. b_n in theirs) of the form $Q = (n + \alpha)^2(n + \beta)(n + \gamma)$, i.e. with a total degree of 4 and a term having a power of two. In many other cases the a_n can be written as $\phi_1(n)^2\phi_2(n)\phi_3(n)$ for some functions ϕ_1, ϕ_2, ϕ_3 . The form $(f(n)^2g(n-1)g(n))$ at the numerator of the continued fraction obtained in Step 1 explains why:

1.2.0.3 Step 3: Operate the change of variable $n_0 = n + \alpha$ to get:

$$Q = n_0^2(n_0 - \alpha + \beta)(n_0 - \alpha + \gamma)$$

1.2.0.4 Step 4: Operate the change of variables $\beta_1 = -\alpha + \beta, \gamma_1 = -\alpha + \gamma$ to get:

$$Q = n_0^2(n_0 + \beta_1)(n_0 + \gamma_1)$$

1.2.0.5 Step 5: Operate the change of variable $n_0 = n_1(\gamma_1 - \beta_1)$ to get:

$$Q = (n_1(\gamma_1 - \beta_1))^2(n_1(\gamma_1 - \beta_1) + \beta_1)(n_1(\gamma_1 - \beta_1) + \gamma_1)$$

1.2.0.6 Step 6: Kick the parasite factor $(\gamma_1 - \beta_1)^2$ out of the continued fraction and integrate it in the limit. We get:

$$Q' = \frac{Q}{(\gamma_1 - \beta_1)^2} = n_1^2(n_1(\gamma_1 - \beta_1) + \beta_1)(n_1(\gamma_1 - \beta_1) + \gamma_1)$$

1.2.0.7 Step 5: Declare $f(n_1) = \text{ID}(n_1) = n_1$ and $g(n_1) = n_1(\gamma_1 - \beta_1) + \beta_1$ to get:

$$Q' = f(n_1)^2g(n_1)(n_1(\gamma_1 - \beta_1) + \gamma_1)$$

1.2.0.8 Step 7: Observe that:

$$g(n_1 - 1) = (n_1 - 1)(\gamma_1 - \beta_1) + \gamma_1 = (\gamma_1 - \beta_1)n_1 + \beta_1$$

and hence:

$$Q' = f(n_1)^2 g(n_1) g(n_1 - 1)$$

1.2.0.9 Step 8: Replace in the simplified continued fraction the terms of the form:

$$f(n)g(n) + f(n-1)g(n-2)$$

by their variable-changed expression in n_1 .

1.2.0.10 Step 9: Light an altar candle hoping that the variable changes did not alter the initial conditions. If so a new relation of the form $a_n = (n + \alpha)^2(n + \beta)(n + \gamma)$ and $b_n = \text{polynomial}$ was found.

Remark 14. Non Polynomial Rational Fractions: Nothing in the preamble assumed that f and g are polynomials or rational fractions, hence any functions satisfying the few properties announced can be used to derive “magic” continued fractions provided that there is a way to write the infinite sum under a fancy closed form. We give a few examples in the next section.

1.3 Implementation

The implementation assumes that $f(0) = 0$ to enforce the initial conditions², sets:

$$a_n = -f(n)^2 \text{ and } b_n = \frac{f(n+1)g(n+1) + f(n)g(n-1)}{g(n)}$$

and prints:

$$L = g(0)^2 \sum_{i=0}^{\infty} \frac{1}{f(i+1)g(i)g(i+1)}$$

As well as the approximate numerical values of both the exact expression and the continued fraction (to visually compare both).

For the sake of compactness we display L and not its inverse.

The code takes f and g from an example list `Ex` into which the reader can plug any desirable function to generate new relations at wish. In the listing above we changed the order of the printed formulae to fit the longest examples in a landscape layout. In the formulae C stands for Catalan’s constant.

²It is also possible to tweak the code to work with other initial conditions provided that $q_{-1}p_{-2} - q_{-2}p_{-1} = 1$, we did not do this here.

```

1
2 Ex={{z^4,1+2z},{z^3,z+1},{z^7,z+1},{z^4/(z+2),(z+3)},{z^5(z+1),z+1},{z^6/(z+2),(z+1)/(z^2+1)},{z^4(z
+1)/(z+2),(z+1)/(z^2+1)},{z^2,(1+Sqrt[2](z^2+z))},{z^4,(1-Sqrt[2](z^2+z))},{z^6,(1+29(z^2+z))},{z
^5,(1+4(z^2+z))},{E^(-8-2z)z^3,E^z},{E^(-8-2z)z(2+5z+2z^2)^2,E^z},{E^(-2-2z)z(1+z)^2,E^z},{E
^(-2z)z(2+z)^2,E^z},{E^(-2-2z)z(1+4z)^2,E^z},{E^(-2z)z(z+z^2)^2,E^z},{E^(-2z)z(1+2z+z^2)^2,E^z
},{E^(-2-2z)z(z+2z^2)^2,E^z},{E^(-2z)z(3z+4z^2)^2,E^z},{9z/Exp[z],Exp[z]},{(-2-3z)z,(2+z)
^10},{(-2-2z)z,(2+z)^10},{(-3-z)z,(2+z)^10},{-2z,(2+z)^10},{z(-2+3z),(2+z)^10},{3z^2,(2+z)^10},{z
(1+3z),(2+z)^10},{(2z+1)/E^z,E^z},{(2z+3)/E^z,E^z},{(2z^2+z)/E^z,E^z},{(2z^2+3z+1)/E^z,E^z},{z
^2,1+z},{z(1+2z),1+z},{z(1+3z),1+z},{z(2+z^2),2+z},{z(2+z+z^2),3+z},{z(2+2z+z^2),4+z},{z(2+4z+z
^2),3+z},{z(4+2z^2),2+z},{z(z+3z^2),1+z},{z(2z+3z^2),1+z},{z(4z+3z^2),1+z},{z(3z+4z^2),1+z},{z(3z
+4z^2),2+z}};
3
4 F:=Function[{w,x},x[[1]]/.{z->w}];
5 G:=Function[{w,x},x[[2]]/.{z->w}];
6 For[i=1,i<=Length[Ex],
7 v=.;
8 CF=(F[v+1,Ex[[i]]] G[v+1,Ex[[i]]]+F[v,Ex[[i]]] G[v-1,Ex[[i]]])/G[v,Ex[[i]]];
9 st=N[{1,0,CF/.v->0,1},10000];
10 For[n=1,n<= 1000,n++,
11 bn=CF/.v->n;
12 an=-F[n,Ex[[i]]]^2;
13 st={{0,0,1,0},{0,0,0,1},{an,0,bn,0},{0,an,0,bn}}.st];
14 formalexpr=G[0,Ex[[i]]]^2/F[t+1,Ex[[i]]]/G[t,Ex[[i]]]/G[t+1,Ex[[i]]];
15 closedform=Simplify[Sum[formalexpr, {t,0,Infinity}]];
16 approxform=N[st[[4]]/st [[3]],1000];
17 Print[{closedform,N[closedform,100],N[approxform,100]};
18 i++]

```

$$\begin{aligned}
& \left\{ v^4, 2v+1, 8 - \frac{2\pi^2}{3} - \frac{\pi^4}{90} \right\} \\
& \left\{ v^3, v+1, -\zeta(3) + \frac{\pi^4}{90} + \frac{\pi^2}{6} - 1 \right\} \\
& \left\{ v^7, v+1, -\zeta(3) - \zeta(5) - \zeta(7) + \frac{\pi^8}{9450} + \frac{\pi^6}{945} + \frac{\pi^4}{90} + \frac{\pi^2}{6} - 1 \right\} \\
& \left\{ \frac{v^4}{v+2}, v+3, \frac{1}{270} (-270\zeta(3) + 9\pi^4 + 15\pi^2 - 55) \right\} \\
& \left\{ v^5(v+1), v+1, -4\zeta(3) - 2\zeta(5) + \frac{\pi^6}{945} + \frac{\pi^4}{30} + \pi^2 - 7 \right\} \\
& \left\{ \frac{v^6}{v+2}, \frac{v+1}{v^2+1}, 11\zeta(3) + 10\zeta(5) + 4\zeta(7) - \frac{5\pi^2}{3} - \frac{11\pi^4}{90} - \frac{2\pi^6}{315} + 10 \right\} \\
& \left\{ \frac{v^4(v+1)}{v+2}, \frac{v+1}{v^2+1}, 4(5\zeta(3) + \zeta(5) + 13) - \frac{41\pi^2}{6} - \frac{\pi^4}{9} \right\} \\
& \left\{ v^2, \sqrt{2}(v^2+v)+1, \frac{\pi^2(\sqrt{2}-4) - 30\sqrt{2} + 36 + 3\pi(3\sqrt{2}-2)\sqrt{2\sqrt{2}-1}\tanh\left(\frac{1}{2}\sqrt{2\sqrt{2}-1}\pi\right)}{6(\sqrt{2}-4)} \right\} \\
& \left\{ v^4, 1-\sqrt{2}(v^2+v), -4\sqrt{2} + \frac{\pi^4}{90} + \frac{\pi^2(3\sqrt{2}+5)}{6\sqrt{2}+3} - 5 - \frac{(27\sqrt{2}+38)\pi\tan\left(\frac{1}{2}\sqrt{2\sqrt{2}+1}\pi\right)}{2(2\sqrt{2}+1)^{3/2}} \right\}
\end{aligned}$$

$$\begin{aligned}
& \left\{ v(v^2 + 2v + 2), v + 4, \frac{49}{18} - \frac{4}{5}\pi \coth(\pi) \right\} \\
& \left\{ v(v^2 + 4v + 2), v + 3, \frac{1}{8} \left(9\sqrt{2}\pi \cot(\sqrt{2}\pi) - 10 \right) \right\} \\
& \left\{ v(2v^2 + 4), v + 2, \frac{1}{12} \left(6 - \sqrt{2}\pi \coth(\sqrt{2}\pi) \right) \right\} \\
& \left\{ v(3v^2 + v), v + 1, \zeta(3) - \frac{2\pi^2}{3} - \frac{9\sqrt{3}\pi}{4} + 40 + \log\left(\frac{1}{3486784401\sqrt[4]{3}}\right) \right\} \\
& \left\{ v(3v^2 + 2v), v + 1, \frac{1}{48} \left(-3(-8\zeta(3) - 65 + \log(3) + 16\log(243)) + 27\pi\sqrt{3} - 10\pi^2 \right) \right\} \\
& \left\{ v(3v^2 + 4v), v + 1, \frac{1}{768} \left(192\zeta(3) - 56\pi^2 + 54\pi\sqrt{3} - 447 - 324\log(2) + 162\log(3) + 324\log(6) \right) \right\} \\
& \left\{ v(4v^2 + 3v), v + 1, \frac{1}{162} \left(54\zeta(3) - 21\pi^2 + 192\pi + 350 - 384\log(8) \right) \right\} \\
& \left\{ v(4v^2 + 3v), v + 2, \frac{1}{270} \left(30\pi^2 - 768\pi - 1049 + 1536\log(8) \right) \right\} \\
& \left\{ e^{-v}(2v^2 + 3v + 1), e^v, e \left(-1 + e(\log(e-1) - 1) + 2\sqrt{e} \tanh^{-1}\left(\frac{1}{\sqrt{e}}\right) \right) \right\} \\
& \left\{ v^2, v + 1, \zeta(3) - \frac{\pi^2}{6} + 1 \right\} \\
& \left\{ v(2v + 1), v + 1, \frac{\pi^2}{6} - 7 + \log(256) \right\} \\
& \left\{ v(3v + 1), v + 1, \frac{1}{12} \left(2\pi^2 + 9\pi\sqrt{3} - 156 + 81\log(3) \right) \right\} \\
& \left\{ v(v^2 + 2), v + 2, 1 - \frac{\pi \coth(\sqrt{2}\pi)}{3\sqrt{2}} \right\} \\
& \left\{ v(v^2 + v + 2), v + 3, \frac{25}{16} - \frac{9\pi \tanh\left(\frac{\sqrt{7}\pi}{2}\right)}{8\sqrt{7}} \right\} \\
& \left\{ e^{-v}(2v + 1), e^v, e \left(\sqrt{e} \tanh^{-1}\left(\frac{1}{\sqrt{e}}\right) - 1 \right) \right\} \\
& \left\{ e^{-v}(2v + 3), e^v, \frac{1}{3}e \left(-3e - 1 + 3e^{3/2} \tanh^{-1}\left(\frac{1}{\sqrt{e}}\right) \right) \right\} \\
& \left\{ e^{-v}(2v^2 + v), e^v, -e \left(-3 + \log(e-1) + 2\sqrt{e} \tanh^{-1}\left(\frac{1}{\sqrt{e}}\right) \right) \right\} \\
& \left\{ v^6, 29(v^2 + v) + 1, \frac{\pi^6}{945} + \frac{87\pi^4}{10} + \frac{306124\pi^2}{3} - \frac{478731681}{2} + \frac{15895741}{10}\pi\sqrt{29}\tan\left(\frac{5\pi}{2\sqrt{29}}\right) \right\} \\
& \left\{ v^5, 4(v^2 + v) + 1, 8\zeta(3) + \zeta(5) + 56 - 48\log(4) \right\} \\
& \left\{ e^{-2v-8}v^3, e^v, e^9\zeta(3) \right\} \\
& \left\{ e^{-2v-8}v(2v^2 + 5v + 2)^2, e^v, -\frac{1}{108}e^9(13\pi^2 + 8(\log(16) - 19)) \right\} \\
& \left\{ e^{-2v-2}v(v+1)^2, e^v, -\frac{1}{6}e^3(\pi^2 - 12) \right\} \\
& \left\{ e^{-2v}v(v+2)^2, e^v, e - \frac{e\pi^2}{12} \right\}
\end{aligned}$$

$$\begin{aligned}
& \left\{ e^{-2v-2} v(4v+1)^2, e^v, -\frac{1}{4} e^3 (8C + \pi^2 + 2\pi + 4(\log(8) - 8)) \right\} \\
& \left\{ e^{-2v} v (v^2 + v)^2, e^v, e \left(\zeta(3) - \frac{\pi^2}{2} + 4 \right) \right\} \\
& \left\{ e^{-2v} v (v^2 + 2v + 1)^2, e^v, -\frac{1}{90} e (90(\zeta(3) - 4) + \pi^4 + 15\pi^2) \right\} \\
& \left\{ e^{-2v-2} v (2v^2 + v)^2, e^v, -\frac{1}{3} e^3 (-3(\zeta(3) + 32) + 5\pi^2 + 36 \log(4)) \right\} \\
& \left\{ e^{-2v} v (4v^2 + 3v)^2, e^v, -\frac{1}{243} e (-288C - 27\zeta(3) + 48\pi^2 - 72\pi - 256 + 144 \log(8)) \right\} \\
& \left\{ 9e^{-v} v, e^v, \frac{1}{9} (e - e \log(e - 1)) \right\}
\end{aligned}$$

$$\left\{ v^1, \sqrt{2} \left(v^2 + v \right) + 2, \frac{4\pi^4 \left(\sqrt{2} - 8 \right) + 30\pi^2 \left(17\sqrt{2} - 12 \right) + 45 \left(64 - 30\sqrt{2} \right) - 90\pi \left(5\sqrt{2} - 11 \right) \sqrt{4\sqrt{2} - 1} \tanh \left(\frac{1}{2} \sqrt{4\sqrt{2} - 1} \pi \right)}{360 \left(\sqrt{2} - 8 \right)} \right\}$$

$$\left\{ v^6, \sqrt{2} \left(v^2 + v \right) + 2, \frac{16\pi^6 \left(\sqrt{2} - 8 \right) + 84\pi^4 \left(17\sqrt{2} - 12 \right) - 6615 \left(19\sqrt{2} - 28 \right) + 630\pi^2 \left(39\sqrt{2} - 64 \right) + 1880\pi \left(21\sqrt{2} - 29 \right) \sqrt{4\sqrt{2} - 1} \tanh \left(\frac{1}{2} \sqrt{4\sqrt{2} - 1} \pi \right)}{15120 \left(\sqrt{2} - 8 \right)} \right\}$$

$$\left\{ v^8, \sqrt{2} \left(v^2 + v \right) + 2, \frac{16\pi^8 \left(\sqrt{2} - 8 \right) + 80\pi^6 \left(17\sqrt{2} - 12 \right) + 22050\pi^2 \left(19\sqrt{2} - 28 \right) + 420\pi^4 \left(39\sqrt{2} - 64 \right) - 14175 \left(149\sqrt{2} - 200 \right) + 9450\pi \left(69\sqrt{2} - 91 \right) \sqrt{4\sqrt{2} - 1} \tanh \left(\frac{1}{2} \sqrt{4\sqrt{2} - 1} \pi \right)}{151200 \left(\sqrt{2} - 8 \right)} \right\}$$

$$\left\{ v^{10}, 1 - \sqrt{2} \left(v^2 + v \right), \frac{2\pi^6 \left(8\sqrt{2} + 11 \right)}{135 \left(4\sqrt{2} + 9 \right)} + \frac{\pi^{10}}{93555} + \frac{\pi^8 \left(13\sqrt{2} + 17 \right)}{4725 \left(4\sqrt{2} + 9 \right)} + \frac{8\pi^4 \left(30\sqrt{2} + 43 \right)}{45 \left(4\sqrt{2} + 9 \right)} - \frac{2\pi^2 \left(176\sqrt{2} + 249 \right)}{4\sqrt{2} + 9} - \frac{2 \left(2159\sqrt{2} + 3057 \right) \pi \tanh \left(\frac{1}{2} \sqrt{2\sqrt{2} + 1} \pi \right)}{\left(2\sqrt{2} + 1 \right)^{5/2}}$$

$$\left\{ v^{12}, \sqrt{2} \left(v^2 + v \right) + 2, \frac{88448\pi^{12} \left(\sqrt{2} - 8 \right) + 436800\pi^{10} \left(17\sqrt{2} - 12 \right) + 76575600\pi^6 \left(19\sqrt{2} - 28 \right) + 2162160\pi^8 \left(39\sqrt{2} - 64 \right) + 170270100\pi^4 \left(149\sqrt{2} - 200 \right) + 425675250\pi^2 \left(1381\sqrt{2} - 1996 \right) - 638512875 \left(4479\sqrt{2} - 6320 \right) + 12770257550\pi \left(689\sqrt{2} - 963 \right) \sqrt{4\sqrt{2} - 1} \tanh \left(\frac{1}{2} \sqrt{4\sqrt{2} - 1} \pi \right)}{81729648000 \left(\sqrt{2} - 8 \right)} \right\}$$

$$\left\{ (-3v - 2)^{10}, (v + 2)^{10}, -\frac{3}{4} \frac{(654616480\zeta(3) + 342927872\zeta(5) + 73113600\zeta(7) + 6422528\zeta(9) + 31737050334 - 2324522934 \log(2) + 1162261467 \log(3) + 2324522934 \log(6))}{4} - \frac{1}{4} \frac{156065456\pi^4}{5} - \frac{913408\pi^8}{225} - \frac{141308416\pi^6}{315} - \frac{131072\pi^{10}}{10395} \right\}$$

$$\left\{ (-2v - 2)^{10}, (v + 2)^{10}, -\frac{512 \left(-467775(4341764\zeta(3) + 1794064\zeta(5) + 389184\zeta(7) + 41216\zeta(9) + 1024\zeta(11) + 110374897) + 48640\pi^{10} + 10638144\pi^8 + 1035033120\pi^6 + 68103341460\pi^4 + 4803747223275\pi^2 \right)}{467775} \right\}$$

$$\left\{ (-v - 3)^{10}, (v + 2)^{10}, \frac{512 \left(-39106972899225 + 3428161436700\pi^2 + 47204776980\pi^4 + 649503360\pi^6 + 5144832\pi^8 + 10240\pi^{10} \right)}{467775} \right\}$$

$$\left\{ v(3v - 2)^{10}, (v + 2)^{10}, -\frac{3(224082559223414400\zeta(3) + 902645353815040000\zeta(5) + 178178187264000000\zeta(7) + 13985382400000000\zeta(9) + 12699985214834440651 - 2324522934 \log(2) - 1162261467 \log(3) - 2324522934 \log(6))}{40000000000} + \frac{05536\pi^{10}}{22275} + \frac{88928768\pi^8}{4000000000} + c \right\}$$

where: $c = \frac{162966967472\pi^6}{984375} + \frac{1113600243258567\pi^4}{937500} + \frac{85944691048496267\pi^2}{1000000000}$

$$\left\{ 3v^2 (v+2)^{10}, \frac{256 \left(-467776(3325728\zeta(3) + 1342656\zeta(5) + 267264\zeta(7) + 21504\zeta(9) + 159629723) + 25600\pi^{10} + 11087312\pi^8 + 131868000\pi^6 + 93801403080\pi^4 + 6769898635500\pi^2 \right)}{1403325} \right\}$$

$$\left\{ s(3v+1)(v+2)^{10}, \frac{512 \left(-1871(-2(1307738292800\zeta(3) + 52934248000\zeta(5) + 10636800000\zeta(7) + 880000000\zeta(9) + 55119052506237 + 11622614671\log(6)) + 2324522934\log(2) - 11022614671\log(3)) + 7249024769679\pi\sqrt{3} + c \right)}{6098203125} \right\}$$

where: $c' = -800000000\pi^{10} - 31767150000\pi^8 - 3702814632000\pi^6 - 2609348019389000\pi^4 - 187811501659121340\pi^2$

$$\left\{ -2v(v+2)^{10}, -3072(106154\zeta(5) + 42664\zeta(7) + 8352\zeta(9) + 640\zeta(11) + 10269357) + \frac{262144\pi^{10}}{31185} + \frac{18661376\pi^8}{225} + \frac{950272\pi^6}{35} + \frac{581261312\pi^4}{15} + 281420392\pi^2 \right\}$$

1.4 Using Formal Manipulations

Mathematica is powerful enough to take into account parametric summations. If we remove from our code the numerical part and add parameters to the input we can get general forms in which we later set parameters at wish.

In the following example we arbitrarily broke the result into several pieces to fit in the page:

```

1 Ex={{(a z)^4+(c z)^3}u,d+b(2+2z)};
2 F:=Function[{w,x},x[[1]]/.{z->w}];
3 G:=Function[{w,x},x[[2]]/.{z->w}];
4 For[i=1,i<=Length[Ex],v=.;
5 CF=(F[v+1,Ex[[i]]]G[v+1,Ex[[i]]]+
6 F[v,Ex[[i]]]G[v-1,Ex[[i]]])/G[v,Ex[[i]]];
7 formalexpr=
8 G[0,Ex[[i]]]^2/F[t+1,Ex[[i]]]/G[t,Ex[[i]]]/G[t+1,Ex[[i]]];
9 closedform=Simplify[Sum[formalexpr,{t,0,Infinity}]];
10 Print[{closedform}];
11 i++]

```

$$\begin{aligned}
 \text{result} &= \frac{\ell_1 + \ell_2 + \ell_3 + \ell_4 + \ell_5 + \ell_7 + \ell_6}{6c^9 d^3 u (2b+d)^2 (2bc^3 - a^4 d) (2bc^3 - a^4 (2b+d))} \\
 \ell_1 &= 96b^3 c^9 (2b+d) \psi^{(0)} \left(\frac{d}{2b} + 1 \right) \left(bc^3 (4b^2 + 6bd + 3d^2) - 2a^4 (2b^3 + 4b^2 d + 3bd^2 + d^3) \right) \\
 \ell_2 &= +6a^{16} d^3 (2b+d)^4 \psi^{(0)} \left(\frac{c^3}{a^4} + 1 \right) + (2bc^3 - a^4 d) \\
 \ell_3 &= 384\gamma b^6 c^6 (c^3 - a^4) - 32b^5 c^3 d (6\gamma a^8 + (24\gamma - \pi^2) a^4 c^3 + (\pi^2 - 24\gamma) c^6) + a^4 d^6 (-6\gamma a^8 + \pi^2 a^4 c^3 - 6c^6 \zeta(3)) \\
 \ell_4 &= -16b^4 d^2 (6\gamma a^{12} + (24\gamma - \pi^2) a^8 c^3 + 2a^4 c^6 (3\zeta(3) - 2\pi^2 + 18\gamma) - 2c^9 (3\zeta(3) - 2\pi^2 + 18\gamma)) \\
 \ell_5 &= 8b^3 d^3 (-24\gamma a^{12} + 4(\pi^2 - 9\gamma) a^8 c^3 - 6a^4 c^6 (4\zeta(3) - \pi^2 + 4\gamma) + c^9 (18\zeta(3) - 5\pi^2 + 18\gamma - 6)) \\
 \ell_6 &= -2bd^5 (24\gamma a^{12} + 2(3\gamma - 2\pi^2) a^8 c^3 - a^4 c^6 (\pi^2 - 24\zeta(3)) - 6c^9 \zeta(3)) \\
 \ell_7 &= -8b^2 d^4 (18\gamma a^{12} + 3(4\gamma - \pi^2) a^8 c^3 + a^4 c^6 (18\zeta(3) - 2\pi^2 + 3\gamma) + c^9 (\pi^2 - 9\zeta(3)))
 \end{aligned}$$

This is much more practical than the blind exploration of values. Whilst the code does nothing with CF, we left it in the program to explicit the continued fraction being calculated. The chosen example has nothing fundamental or conceptual in it and was chosen at random for illustrative purposes.

1.5 Conclusion & Further Challenges

Given the above it appears that several of the Ramanujan Project conjectures³ can be explained and/or automatically machine-proved with virtually no effort.

We did not machine-prove the online conjectures one by one as this implies retyping their polynomials, a retro-solving for f and g by identification and computing the closed forms using formal summation to infinity. Nonetheless, many such relations can be generated automatically quasi-instantaneously on a simple PC.

An interesting question is that of reversing continued fractions from a target constant. For instance, determine a_n and b_n such that the continued fraction converges to a constant chosen *a priori*, e.g.:

³We did not exhaust all the relations listed online.

$$\frac{1}{L} = \sum_{i=2}^{100} \zeta(i)$$

We did not research this challenge and leave it to readers interested in pursuing this line of investigation.

2 Pattern Recognition Experiments on Mathematical Expressions

Based on common work with David Naccache.

2.1 Introduction

Pattern recognition is a process that involves identifying rules in data and matching them with particular case information. Pattern recognition can be seen as a type of machine learning, as it uses machine learning algorithms to recognize patterns in data. This process is characterized by the ability to learn from data, recognize familiar patterns, and recognize patterns even if they are partially visible.

Very schematically, there are three main types of pattern recognition heuristics: statistical pattern recognition, syntactic pattern recognition, and neural pattern recognition.

- *Statistical pattern recognition* involves using particular case data to learn from examples and generalize rules to new observations.
- *Syntactic pattern recognition* (a.k.a structural pattern recognition), involves identifying patterns based on simpler sub-patterns called primitives. For example, opcodes can be seen as primitives that connect to form programs.
- *Neural pattern recognition* relies on artificial neural networks, which are made up of many simple processors and their connections. These networks can learn complex nonlinear input-output relationships and adapt to data through sequential training procedures.

Most pattern recognition heuristics proceed by two steps:

- An *Explorative Stage* that seeks to identify patterns
- A *Descriptive Stage* that categorizes patterns found during exploration

In this work we provide the results of the explorative stage of syntactic pattern recognition on mathematical expressions. Given the nature of the objects we work on (conjectures) the descriptive stage is left to a human.

We give a few examples of conjectured results. None of which was thoroughly checked for novelty. We did not attempt to prove all the relations found and focused on their generation.

2.2 The Pattern Recognition Algorithm

The pattern recognition algorithm has two components called the *generalizer* and the *identifier*.

The generalizer departs from a known continued fraction or a mathematical expression (a particular case) and automatically parameterizes parts of it. The parameterized parts are *target ingredients* tagged by the user. For each set of particular parameter values (taken over search space), approximated values of the formula are collected for later analysis.

Target ingredients are replaced by progressions, denoted by $\mu_{\mathbf{u}}(i)$, which can be constant, (alternating) arithmetic, geometric, harmonic or exponential depending on the parameter choices. Those are captured by the general formula:

$$\mu_{\mathbf{u}}(i) = u_4 i^{u_5} + (u_0 + i u_1)^{u_3} u_2^i$$

For instance, the Ramanujan Machine Project [Raayoni, 2021; David, 2021; Raayoni, 2019] re-discovered an already known relation involving e^π . Namely, that the continued fraction defined by $b_n = n^2 + 4$ and $a_n = 2n + 1$ converges to:

$$\frac{2(e^\pi + 1)}{e^\pi - 1} = 1 + \frac{1^2 + 4}{3 + \frac{2^2 + 4}{5 + \frac{3^2 + 4}{7 + \frac{4^2 + 4}{9 + \ddots}}}}$$

A natural tagging query of this identity for search by the user might hence be:

$$Q(\mathbf{u}) = \mu_{\mathbf{u}}(0) + \frac{\mu_{\mathbf{v}}(0)}{\mu_{\mathbf{u}}(1) + \frac{\mu_{\mathbf{v}}(1)}{\mu_{\mathbf{u}}(2) + \frac{\mu_{\mathbf{v}}(2)}{\mu_{\mathbf{u}}(3) + \frac{\mu_{\mathbf{v}}(3)}{\mu_{\mathbf{u}}(4) + \ddots}}}}$$

With

$$\mathbf{u} = \{\mathbb{Q}, \mathbb{Q}, 1, 1, 0, 0\} \text{ and } \mathbf{v} = \{\mathbb{Z}, 0, 1, 1, \mathbb{Q}, \mathbb{N}\}$$

That is:

$$\mu_{\mathbf{u}}(i) = (\mathbb{Q} + i\mathbb{Q}) \text{ and } \mu_{\mathbf{v}}(i) = \mathbb{Q}i^{\mathbb{N}} + \mathbb{Z}$$

When this is done, the program varies the progressions' parameters over the chosen search spaces and collects sequences of resulting values. The tests that we list here are of course non limitative and many other variants can be added to the proposed heuristic.

Remark 15. Obviously, we are quickly limited by the increasing complexity due to nested loops running over the parameters of the expressions (i.e. the u_i s).

Remark 16. At the risk of overlooking some gold nuggets, when we explore \mathbb{Q} we start by exploring \mathbb{N} and if the search is conclusive, we refine it by increments of $1/6$ which have the advantage of exploring units, halves and thirds at the cost of a small multiplicative factor of 6. If interesting results are found with increments of 6 the step is refined to $1/30$ and to Farey sequences.

The sequences obtained by varying those parameters are fed into the identifier for possible recognition. To detect conjectures the identifier performs a number of tests on the obtained sequences. Tests belong to two categories: *morphological tests* and *serial tests*. Morphological tests are applied to very few individual results and try to spot their characteristics. Serial tests are applied to more results and seek to discover relationships between them.

Algebraic number identification (ANI): Collect 10 convergence limits Q_0, Q_1, \dots, Q_9 and, using LLL [Lenstra, 1982], check if any of those Q_i s is the root of a small degree (≤ 10) polynomial. If so, check that RNI failed before returning true to avoid multiple alerts as rationals are also algebraic. This is a morphological test.

The degree 10 was chosen arbitrarily and can be changed at wish (provided that the precision is matched to the degree).

Rational number identification (RNI): Collect 10 convergence limits Q_0, Q_1, \dots, Q_9 and, using LLL, check if any of those Q_i s is a good approximation of a rational number having a (abnormally) small numerator and a small denominator. This is a morphological test.

Constant presence identification (CPI): Collect 10 convergence limits Q_0, Q_1, \dots, Q_9 . Consider the 45 pairs P_1, P_2 formed from those Q_i s. Using LLL, check the assumption that there is at least one pair of the form:

$$P_1 = \frac{a_1 + b_1 U}{c_1 + d_1 U} \text{ and } P_2 = \frac{a_2 + b_2 U}{c_2 + d_2 U}$$

Where $U \notin \mathbf{Q}$ and $a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2 \in \mathbf{Z}$.

Solving for U and equating we get:

$$a_2 b_1 - a_1 b_2 + (b_2 c_1 - a_2 d_1) P_1 + (a_1 d_2 - b_1 c_2) P_2 + (c_2 d_1 - c_1 d_2) P_1 P_2 = 0$$

Hence, when called with on input 1, $P_1, P_2, P_1 P_2$ LLL will return an abnormally short vector if the coefficients are small (as is usually the case in remarkable identities). This is a morphological test.

Constant to exponent identification (CEI): Collect 10 convergence limits Q_0, Q_1, \dots, Q_9 . Consider the 7 quadruples P_1, P_2, P_3, P_4 formed by successive Q_i s⁴.

Here we assume that at successive ranks the limits are of the form:

$$P_k = \frac{a_k + b_k U^k}{c_k + d_k U^k}$$

Which implies that:

$$U^k = \frac{a_k - c_k P_k}{d_k P_k - b_k}$$

It follows that:

$$U = \frac{(a_{k+1} - c_{k+1} P_{k+1})(d_k P_k - b_k)}{(d_{k+1} P_{k+1} - b_{k+1})(a_k - c_k P_k)}$$

$$\frac{(a_{k+3} - c_{k+3} P_{k+3})(d_{k+2} P_{k+2} - b_{k+2})}{(d_{k+3} P_{k+3} - b_{k+3})(a_{k+2} - c_{k+2} P_{k+2})} = \frac{(a_{k+1} - c_{k+1} P_{k+1})(d_k P_k - b_k)}{(d_{k+1} P_{k+1} - b_{k+1})(a_k - c_k P_k)}$$

Let:

$$S_1 = \{P_k, P_{k+1}, P_{k+2}, P_{k+3}\}$$

$$S_2 = \{P_k P_{k+1}, P_k P_{k+2}, P_{k+1} P_{k+2}, P_k P_{k+3}, P_{k+1} P_{k+3}, P_{k+2} P_{k+3}\}$$

$$S_3 = \{P_k P_{k+1} P_{k+2}, P_k P_{k+1} P_{k+3}, P_k P_{k+2} P_{k+3}, P_{k+1} P_{k+2} P_{k+3}\}$$

$$S = S_1 \cup S_2 \cup S_3 \cup \{1, P_k P_{k+1} P_{k+2} P_{k+3}\}$$

⁴namely: $\{0, 1, 2, 3\}, \{1, 2, 3, 4\}, \{2, 3, 4, 5\}, \{3, 4, 5, 6\}, \{4, 5, 6, 7\}, \{5, 6, 7, 8\}, \{6, 7, 8, 9\}$

When called with on input S LLL will return an abnormally short vector (as is usually the case in remarkable identities). This is a morphological test.

Remark 17. Both CPI and CEI can be generalized to detect the presence of multiple unknown constants in an expression (i.e. U_1, U_2, \dots) or even the presence of common constants in different continued fractions. We did not implement this generalization. Following those tests we can compute a numerical approximation of U and attempt to look it up⁵.

Known constant identification (KCI): Let L be the following set of usual constants:

$$L = \{1, \sqrt{\pi}, \pi, \pi^2, \pi^3, \zeta(3), \zeta(5), \zeta(7), \sqrt{e}, e, e^2, e^3, \phi^2, \gamma, G, \ln 2, \ln 3, \ln 5\}$$

Collect 10 convergence limits Q_0, Q_1, \dots, Q_9 . Check using LLL if any of the Q_i is a number of the form:

$$Q_i \sum_j a_j L_j = \sum_j b_j L_i \text{ for } a_1, a_2, \dots, b_1, b_2 \dots \in \mathbb{Z}$$

If the solution only involves 1, a false is returned. Note that as L increases the required precision must also be increased to prevent spotting artefacts. In practice we (manually) select only a subset of L before running the KCI test according to the nature of the constants appearing in the particular case. Note that KCI and CPI can have overlapping responses.

Rational fraction progression (RFP): In this test we seek to see if when all u_i except one (say \bar{u}) are kept constant, the continued fraction's limit $Q(\bar{u})$ is a ratio of two polynomials in \bar{u} with integer coefficients. This is done by a non linear model fit. The fit residuals serve as a measure of the verdict's likelihood. This is a serial test.

Exponential function progression (EFP): In this test we seek to see if when all u_i except one (say \bar{u}) are kept constant, the continued fraction's limit $Q(\bar{u})$ is a function of the form $ba^{\bar{u}}$ with rational coefficients. This is done by a non linear model fit and rationality detection on a, b . The fit residuals serve as a measure of the verdict's likelihood. If $ab = 0$ return false to avoid reporting the same result as the RFP. This is a serial test.

Inverse exponential progression (IEP): In this test we seek to see if when all u_i except one (say \bar{u}) are kept constant, the continued fraction's limit $Q(\bar{u})$ is a function of the form $ba^{1/\bar{u}}$ with rational coefficients. This is done by a non linear model fit and rationality detection on a, b . The fit residuals serve as a measure of the verdict's likelihood. If $ab = 0$ return false to avoid reporting the same result as the RFP. This is a serial test.

Power plus constant progression (PCP): In this test we seek to see if when all u_i except one (say \bar{u}) are kept constant, the continued fraction's limit $Q(\bar{u})$ is a function of the form $b\bar{u}^a + c$ with rational coefficients. This is done by a non linear model fit and rationality detection on a, b, c . The fit residuals serve as a measure of the verdict's likelihood. If $b = 0$ return false to avoid reporting the same result as the RFP. This is a serial test.

Root plus constant progression (RCP): In this test we seek to see if when all u_i except one (say \bar{u}) are kept constant, the continued fraction's limit $Q(\bar{u})$ is a function of the form $b\sqrt[\bar{u}]{\bar{u}} + c$ with rational coefficients. This is done by a non linear model fit and rationality detection on a, b, c . The fit residuals serve as a measure of the verdict's likelihood. If $ab = 0$ return false to avoid reporting the same result as the RFP. This is a serial test.

⁵e.g. on <https://wayback.cecm.sfu.ca/projects/ISC/ISCmain.html>

ANI	RNI	CPI	CEI	KCI	RFP	EFP	IEP	PCP	RCP
✗	✓	✓	✓	✗	✗	✗	✗	✗	✓

Table 3.1: Test results

2.3 Continued Fractions Converging to $2u(e^{u\pi} + 1)/(e^{u\pi} - 1)$

It appears that the relation:

$$\frac{2(e^\pi + 1)}{e^\pi - 1} = 1 + \frac{1^2 + 4}{3 + \frac{2^2 + 4}{5 + \frac{3^2 + 4}{7 + \frac{4^2 + 4}{9 + \ddots}}}}$$

is the first in an infinite family:

$$\frac{2u(e^{u\pi} + 1)}{e^{u\pi} - 1} = 1 + \frac{1^2 + 4u^2}{3 + \frac{2^2 + 4u^2}{5 + \frac{3^2 + 4u^2}{7 + \frac{4^2 + 4u^2}{9 + \ddots}}}}$$

Indeed, (RCP) linear variations in u cause identifiable $O(\sqrt{u})$ variations in the limit. This is because very quickly:

$$\lim_{u \rightarrow \infty} \frac{e^{u\pi} + 1}{e^{u\pi} - 1} = 1$$

This has the somewhat adverse effect of making the RNI positive very quickly as well.

The final form is detected thanks to the CEI test.

2.3.1 By-product:

Because this holds for $u \in \mathbb{C}^*$, we get a few seemingly “mysterious” corollary identities such as:

$$\frac{2(e + 1)}{\pi(e - 1)} = 1 + \frac{1^2 + 4/\pi^2}{3 + \frac{2^2 + 4/\pi^2}{5 + \frac{3^2 + 4/\pi^2}{7 + \frac{4^2 + 4/\pi^2}{9 + \ddots}}}}$$

$$\frac{6 \ln 2}{\pi} = 1 + \frac{1^2 + 4 \ln^2 2 / \pi^2}{3 + \frac{2^2 + 4 \ln^2 2 / \pi^2}{5 + \frac{3^2 + 4 \ln^2 2 / \pi^2}{7 + \frac{4^2 + 4 \ln^2 2 / \pi^2}{9 + \ddots}}}}$$

2.3.2 Implementation

```

1 f[x_, {m_, d_}] := m/(d + x);
2 For[t = 0, t <= 5,
3   den = Table[2 n + 1, {n, 1, 20000}];
4   num = Table[n^2 + (2 t)^2, {n, 1, 20000}];
5   r = 1 + (Fold[f, Last@num/Last@den, Reverse@Most@Transpose@{num, den}]);
6   e = 2 t (1 + (E^Pi)^t)/((E^Pi)^t - 1);
7   Print[{e, 2 n + 1, n^2 + (2 t)^2, N[{r, e}, 20]}];
8   t += 1/2];

```

2.4 Continued Fractions With Numerator $(v+1+n)(v+2+n)(v+3+n)(v+4+n)$

Let a_n and b_n be two nonzero polynomials and consider the continued fraction:

$$Q = \mathcal{K}_{n=1}^{\infty} \left(\frac{a_n}{b_n} \right)$$

Trivially, if $a_k = 0$ for some k , the nested summation stops and we immediately get Q .

In particular $\forall b_n, a_{n,v} = (v+1+n)(v+2+n)(v+3+n)(v+4+n)$ and $v \leq -6$:

$$Q(v) = \mathcal{K}_{n=1}^{\infty} \left(\frac{a_{n,v}}{b_n} \right) = \mathcal{K}_{n=1}^{-v-5} \left(\frac{a_{n,v}}{b_n} \right)$$

For $-5 \leq v \leq -2$, $Q(v) = 0$. It is thus interesting to examine what happens for $v \in \{0, -1\}$.

Because we do not provide a proper peer-reviewed proof of the relations given here we do not claim them to be theorems, they were however intensively machine-checked.

Defining $b_{n,v,c,t} = c((3+v+n)^2 - t)$, denoting $d = \sqrt{c^2 + 4}$ and

$$Q(v, c, t) = \mathcal{K}_{n=1}^{\infty} \left(\frac{a_{n,v}}{b_{n,v,c,t}} \right)$$

it appears that for $v \in \{0, -1\}$ and $t \in \{0, 1\}$:

$$Q(v, c, t) = \frac{t}{2} (d - c) (5v + 8) + \frac{(t-1)(v+3) \left(-3c^3 + 3\sqrt{d^3} + c(v-13) \right)}{c^2(v-2) - (v-3)^2}$$

With:

$$\ell(v, t) = \lim_{c \rightarrow \infty} cQ(v, c, t) = \frac{(t-1)(3+v)(5+v)}{v-2} + t(5v+8)$$

As checked by the following code:

```

1 Union[Flatten[
2   Table[Expand[(-1 + t) (-3 c^3 + 3 Sqrt[(4 + c^2)^3] +
3     c (-13 + v)) (3 + v)]/(-(-3 + v)^2 + c^2 (-2 + v)) +
4     1/2 (-c + Sqrt[4 + c^2]) t (8 + 5 v)] ==
5   Expand[RootApproximant[
6     N[ContinuedFractionK[(1 + X + v)*(2 + X + v)*(3 + X + v)*(4 +
7       X + v), c ((3 + v + X)^2 - t), {X, 1, 1000}], 20]], {t, 0,
8     1}, {v, -1, 0}, {c, 1, 10}]]][[1]]

```

A better readable split-form is given in Table 3.2.

2.4.1 Generating specific radicals

Inspired by [Chan, 2013], one might want to look for $Q(v, c, t)$ featuring only specific radicals. We illustrate how to generate such relations with:

v	t	$a_{n,v}$	$b_{n,v,c,t}$	$Q(v,c,t)$	$\ell(v,t)$
0	0	$(1+n) \times \dots \times (4+n)$	$c(3+n)^2$	$\frac{3(-3c^3+3\sqrt{(c^2+4)^3-13c})}{-2c^2-9}$	$\frac{15}{2}$
0	1	$(1+n) \times \dots \times (4+n)$	$c((3+n)^2-1)$	$4(\sqrt{c^2+4}-c)$	8
-1	0	$n \times \dots \times (3+n)$	$c(2+n)^2$	$\frac{2(-3c^3+3\sqrt{(c^2+4)^3-14c})}{-3c^2-16}$	$\frac{8}{3}$
-1	1	$n \times \dots \times (3+n)$	$c((2+n)^2-1)$	$\frac{3}{2}(\sqrt{c^2+4}-c)$	3

Table 3.2: The conjectured relations.

$$Q(0,c,0) = \prod_{n=1}^{\infty} \left(\frac{(1+n)(2+n)(3+n)(4+n)}{c(3+n)^2} \right) = \frac{-39c-9c^3+9\sqrt{(4+c^2)^3}}{9+2c^2}$$

If $\tau(c^2+4)$ is a square⁶ then $Q(0,c,0)$ contains the radical $\sqrt{\tau}$.

Note that there are several direct ways to efficiently generate specific radicals, e.g. $Q(0,c,0)$ values related to the golden ratio are $Q(0,|q_5(i)|,0)$ where:

$$q_5(i) = \begin{cases} -1 & \text{if } i \leq 1 \\ -3q_5(i-1) - q_5(i-2) & \text{otherwise.} \end{cases}$$

This yields identities such as, e.g. $41Q(0,4,0) = 88Q(0,1,0) - 348$.

Similarly, to provoke the appearance of a $\sqrt{2}$ use:

$$q_2(i) = \begin{cases} 12i+2 & \text{if } i \leq 1 \\ 6q_2(i-1) - q_2(i-2) & \text{otherwise.} \end{cases}$$

Another possibility is to generate the fast increasing sequences:

$$c_{k,i} = \left\lfloor \left(k + \frac{\sqrt{k^2+4}}{2} \right)^{2i+1} \right\rfloor$$

Where $\forall i \in \mathbb{N}$ each k value generates in $Q(0,c_{k,i},0)$ a radical $\sqrt{\tau}$ where τ is the square-free part of $(k^2+4)/4^{k+1 \bmod 2}$ (OEIS A013946).

The exact same observations are also valid for $\{v,t\} = \{0,1\}, \{-1,0\}, \{-1,1\}$.

2.4.2 Further investigations

The above indicates that it might be interesting to study thoroughly the behavior of continued fractions having a a_n of the form:

$$a_{n,v,\kappa} = (v+\kappa n)(v+1+\kappa n)(v+2+\kappa n)(v+3+\kappa n)$$

⁶i.e. a solution of the generalized Pell equation $\tau(c^2+4) = y^2$.

Table 3.3: Radicals $\sqrt{\tau}$ appearing for all $i \in \mathbb{N}$ in $Q(0, c_{k,i}, 0)$ for $1 \leq k \leq 15$.

k	causes the appearance in $Q(0, c_{k,i}, 0)$ of the radical $\sqrt{\tau}$
1, 4, 11	$\sqrt{5}$
2, 14	$\sqrt{2}$
3	$\sqrt{13}$
5	$\sqrt{29}$
6	$\sqrt{10}$
7	$\sqrt{53}$
8	$\sqrt{17}$
9	$\sqrt{85}$
10	$\sqrt{26}$
12	$\sqrt{37}$
13	$\sqrt{173}$
15	$\sqrt{229}$

One such algebraic continued fraction is given in [Chan, 2013], hence it is very likely that more could be discovered by automatic or algebraic exploration.

2.5 Continued Fractions Converging to Polynomial Roots

It is very well known that:

$$\frac{\sqrt{5}-1}{2} = \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \dots}}}}}}}$$

We tag⁷:

$$Q(\mathbf{u}) = 1 + \cfrac{\mu_{\mathbf{u}}(0)}{\mu_{\mathbf{u}}(0) + \cfrac{\mu_{\mathbf{u}}(0)}{\mu_{\mathbf{u}}(0) + \cfrac{\mu_{\mathbf{u}}(0)}{\mu_{\mathbf{u}}(0) + \cfrac{\mu_{\mathbf{u}}(0)}{\mu_{\mathbf{u}}(0) + \cfrac{\mu_{\mathbf{u}}(0)}{\mu_{\mathbf{u}}(0) + \cfrac{\mu_{\mathbf{u}}(0)}{\mu_{\mathbf{u}}(0) + \cfrac{\mu_{\mathbf{u}}(0)}{\mu_{\mathbf{u}}(0) + \dots}}}}}}}$$

With:

$$\mathbf{u} = \{\mathbb{Q}, 0, 1, 1, 0, 0\} \Rightarrow \mu_{\mathbf{u}}(i) = \mathbb{Q}$$

It appears that for $u \in \mathbb{Q}/[-4, 0]$ LLL identifies that the limit is a root of a second degree polynomial, namely:

$$Q(u) = 1 + \cfrac{u}{u + \cfrac{u}{u + \cfrac{u}{u + \cfrac{u}{u + \cfrac{u}{u + \cfrac{u}{u + \cfrac{u}{u + \dots}}}}}}}$$

$$Q(u)^2 + u(Q(u) - 1) = 0$$

Which is trivial to prove by pushing the u into the continued fraction.

The CPI is positive because for $u = 1$ and $u = 5$ the respective values of $Q(u)$ comprise the common value $\sqrt{5}$.

⁷Adding a 1+ by commodity which does not change anything about the infinite convergence.

ANI	RNI	CPI	CEI	KCI	RFP	EFP	IEP	PCP	RCP
✓	✗	✓	✓	✗	✗	✗	✗	✗	✗

Table 3.4: Test results

2.5.1 Implementation

```

1 f[x_, {m_, d_}] := m/(d + x);
2 For[L = -20, L <= 20 ,
3   If[-4 <= L <= 0, L = 2/3];
4   num = den = Table[L, {n, 1, 200}];
5   r = Fold[f, Last@num/Last@den, Reverse@Most@Transpose@{num, den}];
6   Print[{L, N[r^2 + L (r - 1)]};
7   L += 2/3];

```

2.6 Continued Fractions Converging to $e^{2/\kappa}$

The following relations are well-known⁸:

$$e = 2 + \frac{1}{1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{1 + \frac{1}{4 + \frac{1}{1 + \frac{1}{1 + \frac{1}{6 + \dots}}}}}}}}$$

$$\sqrt{e} = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{5 + \frac{1}{1 + \frac{1}{1 + \frac{1}{9 + \frac{1}{1 + \frac{1}{1 + \frac{1}{13 + \dots}}}}}}}}}}$$

$$\sqrt[3]{e} = 1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{1 + \frac{1}{8 + \frac{1}{1 + \frac{1}{1 + \frac{1}{14 + \frac{1}{1 + \frac{1}{1 + \frac{1}{20 + \dots}}}}}}}}}}}}$$

We hence tag the ones as constants, the progression as arithmetic and let the algorithm monitor the evolution of the limits.

Let $b_n = 1$. Define $\mu(u) = \kappa(u + 1/2) - 1$ for $\kappa \in \mathbb{R}$ and:

$$a_n = \begin{cases} \mu(n/3) = \frac{\kappa(2n+3)}{6} - 1 & \text{if } n \bmod 3 \equiv 0 \\ 1 & \text{otherwise.} \end{cases}$$

In other words, a_n is the sequence:

$$a_n = \{\mu(0), 1, 1, \mu(1), 1, 1, \mu(2), 1, 1, \mu(3), 1, 1, \mu(4), 1, 1, \dots\}$$

Then we detect that the continued fraction generated by a_n, b_n converges to $e^{2/\kappa}$.

The CEI is positive because, for instance $(e^{2/\kappa})^2 = e^{2/\kappa'}$ implies that $2/\kappa = \kappa'$ which is satisfied for several pairs of integer values.

⁸<https://link.springer.com/content/pdf/bbm:978-94-91216-37-4/1.pdf>

ANI	RNI	CPI	CEI	KCI	RFP	EFP	IEP	PCP	RCP
✗	✗	✗	✓	✗	✗	✗	✓	✗	✗

Table 3.5: Test results

2.6.1 Implementation

```

1 f[x_, {m_, d_}] := m/(d + x);
2 For[k = -10, k <= 10,
3   phi = Table[k n + k/2 - 1, {n, 0, 2000 - 1}];
4   num = Table[1, {n, 1, 2000}];
5   den = Take[
6     Flatten[Table[{phi[[i]], {1, 1}}, {i, 1, Floor[2000/3] + 1}], {1,
7       2000}];
8   r = 1 + (Fold[f, Last@num/Last@den, Reverse@Most@Transpose@{num, den}]);
9   v = E^(2/k);
10  Print[{k, v, N[{r, v}, 20]};
11  k += 1/2];

```

2.7 Classical Continued Fractions Involving Catalan's Constant

It is well known that:

$$2G = 2 - \frac{1^2}{3} + \frac{2^2}{1} - \frac{2^2}{3} + \frac{4^2}{1} - \frac{4^2}{3} + \frac{6^2}{1} - \frac{6^2}{3} + \frac{8^2}{1} - \frac{8^2}{3} + \dots$$

We define:

$$\Delta(u, v) = \frac{1}{2v} \times \left(\frac{1^2}{u} + \frac{2^2}{v} - \frac{2^2}{u} + \frac{4^2}{v} - \frac{4^2}{u} + \frac{6^2}{v} - \frac{6^2}{u} + \frac{8^2}{v} - \frac{8^2}{u} + \dots \right)$$

For all the following we observe that $\Delta(u, v) = \Delta(v, u)$.

2.7.1 For $u = 1$

An exploration for $\mathbf{u} = \{0, \mathbb{N}, \mathbb{N}, \mathbb{N}, \mathbb{Z}, 0\}$ reveals that for $u_0 = 0, u_1 = 2, u_2 = 1, u_3 = 2, u_4 = -1, u_5 = 0$ we get identities when $v = 4i^2 - 1$ with the convergence values given in Table 3.6:

Where the general formula for $i > 1$ is:

$$\Delta(1, 4i^2 - 1) = (-1)^{i+1} \left(\sum_{k=0}^{i-1} \frac{(-1)^k}{(2k+1)^2} - G \right)$$

2.7.1.1 Implementation

u	i	$v = 4i^2 - 1$	$\Delta(u, 4i^2 - 1) = \Delta(1, 4i^2 - 1)$
1	0	-1	$1 - G$
1	1	3	$-8/9 + G$
1	2	15	$209/225 - G$
1	3	35	$-10016/11025 + G$
1	4	63	$91369/99225 - G$
1	5	99	$-10956424/12006225 + G$
1	6	143	$1863641881/2029052025 - G$

Table 3.6: The first convergence values for $u = 1$

ANI	RNI	CPI	CEI	KCI	RFP	EFP	IEP	PCP	RCP
✗	✗	✓	✗	✓	✗	✗	✗	✗	✗

Table 3.7: Test results

```

1 f[x_, {m_, d_}] := m/(d + x);
2 For[i = 1, i < 40,
3   {u, v} = {1, 4 i^2 - 1};
4   num = Take[
5     Prepend[Flatten[Table[{(2 n)^2, (2 n)^2}, {n, 1, 100000}], 1] ,
6     100000];
7   den = Flatten[Table[{u, v}, {n, 1, 100000/2}]];
8   r = Fold[f, Last@num/Last@den, Reverse@Most@Transpose[{num, den}]]/2/v;
9   val = (-1)^(i + 1) (Sum[(-1)^k/(2 k + 1)^2, {k, 0, i - 1}] - Catalan);
10  Print[{i, v, val, N[{val, r}, 30]}];
11  i++];

```

Remark 18. Note that the denominators of the numbers:

$$\eta(i) = \sum_{k=0}^{i-1} \frac{(-1)^k}{(2k+1)^2}$$

are interesting by their own right. At first sight they might seem perfect squares but in reality some may contain very small prime factors to an odd power.

2.7.2 For $u = 3$

The exploration in this section is interesting. It was done manually but we would have never had the idea to probe in that specific direction without the insight for the case $u = 1$ produced in the previous section.

The sequence $f(i)$ is **nearly** the absolute value of the OEIS sequence A006309⁹:

1, 5, 21, 33, 65, 85, 133, 161, 261, 341, 481, 533, 645, 705, 901, ~~12803~~, 1281,
1541, 1633, 1825, ~~14615~~, ~~11537~~, 2581, 3201, 3333...

⁹<https://oeis.org/A006309>.

u	i	$f(i)$	$\Delta(3, f(i))$
3	0	1	$\Delta(1, -1)$
3	1	5	$\Delta(1, 3)$
3	2	21	$\Delta(1, 35)$
3	3	33	$\Delta(1, 63)$
3	4	65	$\Delta(1, 143)$
3	5	85	$\Delta(1, 255)$

Table 3.8: The first convergence values for $u = 3$

An unexplained phenomenon occurs for the “abnormally larger” OEIS sequence A006309 values 12803, 14615, 11537 that remains unmatched by any $\eta(i)$ value. We do not have an explanation for this phenomenon that requires further research.

2.7.2.1 Implementation

The following implementation was purposely left unoptimized for the sake of clarity. We start by generating the target values for $u = 3$ and store them in an array. Then we re-generate the values for $u = 1$ and match the array’s contents.

```

1 AbsA006309 =
2   Abs[{1, 5, -21, 33, -65, 85, -133, 161, 261, -341, -481, 533, -645,
3     705, 901, -12803, -1281, -1541, 1633, -1825}];
4 t = {};
5 f[x_, {m_, d_}] := m/(d + x);
6 For[i = 1, i <= Length[AbsA006309],
7   {u, v} = {3, AbsA006309[[i]]};
8   num = Take[
9     Prepend[Flatten[Table[{(2 n)^2, (2 n)^2}, {n, 1, 40000}], 1],
10    40000];
11   den = Flatten[Table[{u, v}, {n, 1, 40000/2}]];
12   r = Fold[f, Last@num/Last@den, Reverse@Most@Transpose@{num, den}]/2/
13     v;
14   AppendTo[t, {AbsA006309[[i]], N[r, 30]}];
15   i++;
16
17 For[j = 1, j <= Length[AbsA006309],
18   If[t[[j, 1]] == 12803,
19     Print["Exception, the value 12803 is skipped."],
20     For[i = 1, i <= 1000000,
21       {u, v} = {1, 4 i^2 - 1};
22       den = Flatten[Table[{u, v}, {n, 1, 40000/2}]];
23       r = Fold[f, Last@num/Last@den, Reverse@Most@Transpose@{num, den}]/
24         2/v;
25       val = (-1)^(i + 1) (Sum[(-1)^k/(2 k + 1)^2, {k, 0, i - 1}] -
26         Catalan);
27       If[Abs[t[[j, 2]] - r] < 10^(-6),
28         Print[{i, N[r, 30], N[t[[j, 2]], 30}], "Entry", j, ":", val,

```

u	v	$\Delta(u, v)$
5	7	$\Delta(1, 15)$
5	39	$\Delta(1, 143)$
5	51	$\Delta(1, 255)$
7	9	$\Delta(1, 35)$
9	11	$\Delta(1, 63)$
11	13	$\Delta(1, 99)$
13	15	$\Delta(1, 143)$

Table 3.9: Other convergence values.

```

29     "_matched_with_Delta[3, t[[j, 1]], ""];
30     i = Infinity];
31     i++];
32     j++];

```

2.7.3 Subsequent u values.

Table 3.9 provides some additional examples for various u, v combinations.

2.7.4 Variations in the numerator.

Let, for instance, $(u, v) = (1, 3)$. Removing the $1/(2v)$ factor in Δ and replacing the $(2n)^2$ by $(n - i)^2$ we get convergence to:

$$1, \frac{4}{5}, \frac{31}{51}, \frac{16}{33}, \frac{355}{883}, \frac{11524}{33599}, \frac{171887}{575075}, \frac{10147688}{38326363}, \dots$$

With the limits being quickly reached after a constant number of terms in the continued fraction.

2.7.4.1 Implementation

```

1 For[i = 1, i < 20,
2 f[x_, {m_, d_}] := m/(d + x);
3
4 num = Take[
5   Prepend[Flatten[Table[{(n - i)^2, (n - i)^2}, {n, 1, 400}], 1] ,
6   400];
7 den = Flatten[Table[{1, 3}, {n, 1, 400/2}]];
8 r = (Fold[f, Last@num/Last@den, Reverse@Most@Transpose@{num, den}]);
9 Print[r];
10 i += 1]

```

2.8 Generalized Cloître Series

In an unpublished note [Cloître,], Benoît Cloître gives a beautiful BBP formula for π^2 based on the identity:

i/ℓ	j/ℓ	$1/\ell$
11	5	8
14	6	10
23	9	16
26	10	18
22/5	8/5	3
31	9	20
28	8	18
19	5	12
16	4	10
13	3	8

Table 3.10: Example relations for which $i + j = 2$

i/ℓ	j/ℓ	$1/\ell$
76	16	30
46	10	18
41	9	16
26	6	10
21	5	8

Table 3.11: Example relations for which $i + j \neq 2$

$$\sum_{k=1}^{\infty} \frac{\cos(ik\pi) (2\cos(j\pi))^k}{k^2} = (\ell\pi)^2$$

Here are some i, j, ℓ combinations detected automatically:

A simple rule allowing to generate many identities consists in fixing a fractional step $1/u$, letting $i = \kappa/u$ for $\pi/3 \leq i \leq 2\pi/3$ and calculating the limit for $\{i, j\} = \{\kappa u, 2 - \kappa u\}$ (e.g. Table 3.10). However, limits for which $i + j \neq 2$ exist as well (e.g. Table 3.11).

2.9 Conclusion & further research

The results given in this section show that pattern matching can obviously be of help in detecting new mathematical conjectures. The very basic processes described in the previous sections can be improved and generalized in a number of ways. The first is obviously an enriching of the collection of tests. The second is deeper exploration which is highly dependent on the computational capabilities at hand. Finally the interpretation of results and the early pruning of less probable branches in the potential conjecture tree can also bring efficiency and pertinence in the discovered relations.

ANI	RNI	CPI	CEI	KCI	RFP	EFP	IEP	PCP	RCP
✗	✗	✓	✓	✓	✗	✗	✗	✗	✗

Table 3.12: Test results

3 A Note on Moments of Random Multiplicative Functions and Truncated Characteristic Polynomials

Based on common work with David Naccache.

3.1 Introduction

At section 7 of [Heap, 2015], Heap and Lindqvist provide a conjectured bound:

$$\mathbb{E}(|S_x|) \leq (1 + o(1)) \cdot 0.9036020036 \dots \sqrt{x}$$

To avoid unnecessarily reproducing the context we refer the reader to [Heap, 2015] and include here only novel ideas improving the above bound. This note can hence be read as a continuation of [Heap, 2015].

We denote by p_i the i -th prime.

Note that it is now known [Harper, 2020] that $\mathbb{E}|S_x| \asymp \frac{\sqrt{x}}{\log \log x}$ which reduces the interest of this note to exploring the limits of the Cauchy-Schwarz strategy.

3.2 The improvement strategy

For understanding the improvement strategy we remark the following:

Let $P(X_{\ell(1)}, \dots, X_{\ell(t)})$ be a multivariate polynomial.

Consider the operator **Replace** taking P and substituting all occurrences of $X_{\ell(i)}^2$ by 1 and all occurrences of $X_{\ell(i)}$ by $1/\ell(i)$.

Form the rational fraction F in the variables $v_{\ell(1)}, \dots, v_{\ell(t)}$:

$$F(v_{\ell(1)}, \dots, v_{\ell(t)}) = \frac{\text{Replace} \left(\prod_{i=1}^t (1 - v_{\ell(i)} X_{\ell(i)})^2 \right)}{\prod_{i=1}^t (1 - v_{\ell(i)}^2)}$$

The minimum of F over $[0, 1]^t$ is reached for:

$$\bar{v}_{\ell(i)} = \ell(i) - \sqrt{\ell(i)^2 - 1}$$

To see why, develop first

$$\begin{aligned} \Delta &= \text{Replace} \left(\prod_{i=1}^t (1 - v_{\ell(i)} X_{\ell(i)})^2 \right) \\ &= \text{Replace} \left(\prod_{i=1}^t (1 - 2v_{\ell(i)} X_{\ell(i)} + v_{\ell(i)}^2 X_{\ell(i)}^2) \right) \\ &= \text{Replace} \left(\prod_{i=1}^t (1 - 2v_{\ell(i)} X_{\ell(i)} + v_{\ell(i)}^2) \right) \\ &= \prod_{i=1}^t \left(1 - 2 \frac{v_{\ell(i)}}{\ell(i)} + v_{\ell(i)}^2 \right) \end{aligned}$$

Which gives:

$$F(v_{\ell(1)}, \dots, v_{\ell(t)}) = \prod_{i=1}^t \frac{1 - 2\frac{v_{\ell(i)}}{\ell(i)} + v_{\ell(i)}^2}{1 - v_{\ell(i)}^2}$$

Hence:

$$\frac{\partial F}{\partial v_{\ell(u)}} = \frac{4\ell(u)v_{\ell(u)} - 2(1 + v_{\ell(u)}^2)}{\ell(u)(v_{\ell(u)}^2 - 1)^2} \prod_{i \neq u} \frac{1 - 2\frac{v_{\ell(i)}}{\ell(i)} + v_{\ell(i)}^2}{1 - v_{\ell(i)}^2}$$

And:

$$\frac{\partial F}{\partial v_{\ell(u)}} = 0 \Rightarrow \bar{v}_{\ell(u)} = \ell(u) \pm \sqrt{\ell(u)^2 - 1}$$

Substituting back the second root $\ell(u) - \sqrt{\ell(u)^2 - 1}$ we get:

$$F(\bar{v}_{\ell(1)}, \dots, \bar{v}_{\ell(t)}) = \sqrt{\prod_{i=1}^t \frac{\ell(i)^2 - 1}{\ell(i)^2}} < 1$$

Hence, when $\ell(i) = p_i$ we have that:

$$\lim_{t \rightarrow \infty} F(\bar{v}_{\ell(1)}, \dots, \bar{v}_{\ell(t)}) = \sqrt{\frac{1}{\zeta(2)}} = \frac{\sqrt{6}}{\pi}$$

The above improves the upper bound on the Steinhaus expectation for $k = 1/2$ given in [Heap, 2015].

To see why, let $0 \leq v_{\ell(1)}, \dots, v_{\ell(t)} \leq 1$ and let $S_x = \sum_{n \leq x} X_n$ as in [Heap, 2015]. By the Cauchy–Schwarz inequality we have:

$$\begin{aligned} \mathbb{E}[|S_x|^2] &\leq \mathbb{E}\left[\left|\prod_{i=1}^t (1 - v_{\ell(i)} X_{\ell(i)}) S_x\right|^2\right] \cdot \mathbb{E}\left(\left|\prod_{i=1}^t (1 - v_{\ell(i)} X_{\ell(i)})\right|^{-2}\right) \\ &= \frac{\mathbb{E}\left[\left|\prod_{i=1}^t (1 - v_{\ell(i)} X_{\ell(i)}) S_x\right|^2\right]}{\prod_{i=1}^t (1 - v_{\ell(i)}^2)} \\ &= F(v_{\ell(1)}, \dots, v_{\ell(t)}) \end{aligned}$$

whose minimum was given before.

t	bound	exact expression	comment
2	0.816497	$\sqrt{2/3}$	bound of [Heap, 2015]
3	0.800000	$4/5$	
4	0.791795	$16\sqrt{3}/35$	
5	0.788516	$96\sqrt{2/5}/77$	
6	0.786180	$384\sqrt{3/35}/143$	
7	0.784818	$4608\sqrt{6/35}/2431$	
8	0.783731	$55296\sqrt{3/7}/46189$	
9	0.782989	$663552/(96577\sqrt{77})$	
10	0.782524	$1327104\sqrt{30/11}/2800733$	
\vdots	\vdots	\vdots	
∞	0.779697	$\sqrt{6}/\pi$	

Table 3.13: Convergence to $\sqrt{6}/\pi$

4 A Conjecture From a Failed Cryptanalysis

Based on common work with David Naccache.

4.1 Introduction

During a failed cryptanalysis of multivariate signature scheme we stumbled on the following observation.

Let $P(x, y)$ be a bivariate polynomial with coefficients in \mathbb{C} . Form the $n \times n$ matrices L_n whose elements are defined by $P(i, j)$. Define the matrices $M_n = L_n - \text{ID}_n$.

It appears that $\mu(n) = (-1)^n \det(M_n)$ is a polynomial in n that we did not characterize.

If we replace the definition of μ by $\mu(n) = (-1)^{n+1} \det(M_n)$ then a similar phenomenon occurs with $M_n = L_n + \text{ID}_n$.

We did not research the reasons for this behavior but noted it for those who wish to further investigate it. The conjecture was later proved by Zhang [Zhang, 2022].

4.2 Example

Let

$$P(x, y) = hx^2y + gy^2x + fy^2 + ex^2 + dxy + ax + by + c$$

Then

$$\begin{aligned} \mu(n) &= (-1)^n \det(M_n) = \sum_{i=0}^9 \eta_i n^i \\ \eta_9 &= \frac{def + cgh - afh - beg}{2160} \\ \eta_8 &= -\frac{gh}{240} \\ \eta_7 &= -\frac{eg + fh + gh}{60} - 6\eta_9 \\ \eta_6 &= \frac{ah + bg - de - df}{72} - \frac{4ef}{45} - \frac{7eg + 7fh}{120} - \frac{7gh}{360} \\ \eta_5 &= \frac{ah + bg - de - df - eg - fh}{24} + \frac{cg + ch - af - be}{12} - \frac{ef}{6} + 9\eta_9 \\ \eta_4 &= \frac{cd + eg + fh - ab}{12} + \frac{de + df - ah - bg + 7ef}{36} + \frac{13gh}{720} - \frac{g + h}{4} + \eta_5 - 9\eta_9 \\ \eta_3 &= -\frac{d + e + f}{3} - \frac{g + h}{2} - \eta_5 - \eta_9 - \eta_7 \\ \eta_2 &= \frac{ab - cd}{12} - \frac{a + b + d + e + f}{2} - \frac{ef}{60} - \eta_5 + \eta_6 - 2\eta_7 + 2\eta_8 - 3\eta_9 - \frac{g + h}{4} \end{aligned}$$

$$\eta_1 = -\frac{a+b}{2} - c - \frac{d+e+f}{6}$$

$$\eta_0 = 1$$

The Mathematica code generating those polynomials is very simple:

```
M := Function[n,
  P := Function[{x, y},
    h x^2 y + g y^2 x + f y^2 + e x^2 + d x y + a x + b y + c];
  Table[P[i, j] , {i, 1, n}, {j, 1, n}] - IdentityMatrix[n]]

t = Table[ Det[(-1)^(k) M[k]], {k, 1, 20}];
mu = Collect[Expand[InterpolatingPolynomial[t, n]], n];
```

5 The Balkans Continued Fraction

Based on common work with David Naccache.

5.1 Introduction

In a previous escapade [Naccache, 2023d] we gave a collection of continued fractions involving Catalan’s constant. This section provides

provides more general formulae governing those continued fractions. Having distinguished different cases associated to regions in the plan, we nickname those continued fractions “The Balkans” as they divide into areas which are related but still different in nature.

Because we do not provide formal proofs of those machine-constructed formulae we do not claim them to be theorems. Still, each and every proposed formula was extensively tested numerically.

All the programs included in this article are *self-contained*, i.e. any code snippet can be run independently of the others to fully illustrate the encoded formula. This renders the code longer but has the great advantage of allowing the reader to run and modify each snippet directly by just cutting and pasting it into Mathematica without requiring any other module¹⁰. The code was compacted for the sake of concision but loading it into Mathematica’s editor re-indent it automatically.

The code in this paper is necessary. Because we do not provide proofs explaining the discovered structures, any slight L^AT_EX misprint would be impossible to fix. Hence readers can consider the code as an unambiguously tested version of the proposed formulae.

5.2 Notations

We denote by $n!!$ the semifactorial of, i.e. the product of all the integers from 1 up to n having the same parity as n :

$$n!! = \prod_{k=0}^{\lceil \frac{n}{2} \rceil - 1} (n - 2k) = n(n-2)(n-4) \cdots$$

Because in all the following we will only apply semifactorials to odd numbers, this can be simplified as:

$$n!! = \prod_{k=1}^{\frac{n+1}{2}} (2k-1) = n(n-2)(n-4) \cdots 3 \cdot 1$$

We denote by Catalan’s constant by $G = 0.91596559 \dots$ and let C_n be the n -th Catalan number:

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

The first Catalan numbers are:

$$1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, \dots$$

¹⁰Each snippet ends by a `ClearAll["Global`*"]`; command whose purpose is to make Mathematica “forget” all passed history.

5.3 The target

We define for odd j and $\kappa, c \in \mathbb{N}$ the following quantity nicknamed “*The Balkans continued fraction*”:

$$Q_{j,\kappa,c} = j(2-j+2\kappa) + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n(c+n)(j+n-1)(1-j+2\kappa+n)}{j(2-j+2\kappa) + (3+4\kappa)n + 3n^2} \right)$$

The question asked is that of finding a general process allowing to compute $Q_{j,\kappa,c}$ *without resorting* to numerical simulations or integer relation algorithms. The reason for this is that while integer relation algorithms allow us to “magically” discover relations, they do not provide *general information* about the underlying structure of the constants found. Why? Because we’re in it for the thrill of discovery, not just the magic of shortcuts.

5.3.1 The contribution

The main contribution of this paper is a collection of formulae computing $Q_{j,\kappa,c}$ without requiring any numerical simulation for positive j, κ, c and odd j .

5.3.2 Why this formula in particular?

The Ramanujan Machine Project [Raayoni, 2021; Cohen, 2022], lists a few continued fractions involving G detected in 2020. We do not know why the project did not resort to (rather basic) integer relation algorithms to discover more relations. We hence decided to play detective and unleashed LLL that found a few hundreds of continued fractions involving G in a few intensive calculation days. All continued fractions were of the form:

$$\epsilon + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n(n+\tau)(n+\eta)(n+\mu)}{\epsilon + \delta n + 3n^2} \right) = \frac{a_0}{a_1 + a_2 G} \text{ where } a_0, a_1, a_2 \in \mathbb{Z}$$

We performed two natural tests on the coefficient vectors $(\delta, \epsilon, \tau, \eta, \mu)$: a PCA to determine if the coefficients can be expressed as linear combinations of less than 5 variables and a Hough transform to detect affine relations in the dataset.

PCA revealed that, when projected on $(\delta, \tau, \eta, \mu)$, nearly all data was governed by three linear dimensions¹¹. We hence understood that we were facing a linear behavior in a large region (Balkans) plus some sporadic cases (see Remark 30). This was also confirmed by the Hough transform that detected several parallel planes in the 3D-space. We thus decided to focus our efforts on the main planes in the 3D-space and understand them.

5.3.3 How formulae were reverse-engineered

Now comes the real adventure. The process that allowed us to reverse-engineer the formulae given in this paper is interesting by its own right. A quick look at many examples of the three quantities a_0, a_1, a_2 forming the fractions:

$$Q_{j,\kappa,c} = \frac{a_0}{a_1 + a_2 G} \text{ where } a_0, a_1, a_2 \in \mathbb{Z}$$

¹¹To which we gave the names j, κ, c .

showed that the a_i s are products of small prime factors and a few large prime factors. This suggested that a_i s were initially¹² of some form:

$$a_i = \text{expression}(j, \kappa, c) = \frac{\prod_{i=0}^{u-1} \phi'_i(j, \kappa, c)}{\prod_{i=0}^{v-1} \phi_i(j, \kappa, c)}$$

where the ϕ_i are functions such as $(an + b)!$, 2^{an+b} , $(an + b)!!$, C_{an+b} , Pochhammer symbols of linear combinations of the parameters j, κ, c etc. and a few unknown “mixing” functions causing the appearance of the large prime factors, e.g. polynomials or recurrence relations.

Fortunately, integer relation algorithms allow us to collect many instances of such forms for diverse j, κ, c values. Hence the problem at hand consists in identifying which ϕ_i s are compatible with the cancellations due to the division. If a given ϕ is present in the expression then it is reasonably assumed that when tried for many j, κ, c the new expression:

$$\text{expression}_1(j, \kappa, c) = \frac{\text{expression}(j, \kappa, c)}{\phi(j, \kappa, c)}$$

or

$$\text{expression}_1(j, \kappa, c) = \text{expression}(j, \kappa, c) \cdot \phi(j, \kappa, c)$$

will feature less small factors and hence stand-out as an outlier.

The process can hence be repeated with proper backtracking until all the combinatorial ϕ_i s were peeled-off. Then it remains to detect what the remaining “mixing” functions are which is done by monitoring the average growth rate of those surviving constants to emit hypotheses on the type of recurrence relations (or polynomials) at hand or resorting to a variety of integer sequence recognition tools to identify the hidden culprits.

- We started our exploration with the simplest case of [Bosnia & Herzegovina](#) where $a_2 = 0$. We chose [Bosnia & Herzegovina](#) as a launchpad because it is the simplest *finite* Balkan continued fraction. We (rightfully) hoped that analyzing it will give us insight about the Balkans’ structure before hell breaks loose with infinite continued fractions spitting G s into the convergence values.
- Having reverse-engineered [Bosnia & Herzegovina](#) we moved to [Croatia](#) for which $a_2 = 0$ as well. Because [Croatia](#) is a more complex version of [Bosnia & Herzegovina](#), the insight gained in [Bosnia & Herzegovina](#) proved very helpful to derive further characteristics of [Croatia](#). We thus stress that the distinction between [Bosnia & Herzegovina](#) and [Croatia](#) is paedagogic rather than scientific.
- The insight gained in [Croatia](#) guided our software to the formula for which is simpler than [Kosovo](#) and [Serbia](#) given that [Montenegro](#) corresponds to $j = 1$.
- Having inferred [Montenegro](#) we moved on to [Kosovo](#) whose symmetry¹³ with [Serbia](#) was quickly noted.

This work demonstrates the interest of statistical classifiers such as Maximum Likelihood Estimation (MLE) and Support Vector Machines (SVM) in mathematical exploration.

¹²i.e., before simplification intervenes.

¹³This symmetry simply comes from the fact that the equation $j(2 - 2j + 2\kappa) = x(2 - 2x + 2\kappa)$ has the two solutions $x = j$ and $x = \kappa - j + 1$.

Area	Domain	$Q_{j,\kappa,c}$ is in
Croatia	$j \geq 2\kappa + 5$	\mathbb{Q}
Bosnia & Herzegovina	$j = 2\kappa + 3$	\mathbb{Q}
Serbia	$2\kappa + 1 \geq j \geq \kappa + 3$	\mathbb{R}
Kosovo	$\kappa + 2 \geq j \geq 3$	\mathbb{R}
Montenegro	$j = 1$	\mathbb{R}

Table 3.14: Areas. We list fractions involving G as being in \mathbb{R} although it is currently unknown if $G \in \mathbb{Q}$ (this is a major open question).

As will be shown, this “gradient descent” method proved itself very well, although it required a few thousands of computation hours on a very powerful cluster.

We estimate that 80% of the discovery effort was done by the machine. The remaining 20% being human “piloting” that, we are convinced, is already at the reach of today’s most powerful LLMs, such as Gemini.

It appears much better to read the coming sections first to understand what prey we are stalking and then refer to Section 5.12 describing the hunting process.

5.4 Kosovo, Serbia, Croatia, Montenegro, Bosnia & Herzegovina

As we will see, we distinguish five cases that we call Kosovo, Serbia, Croatia, Montenegro and Bosnia & Herzegovina after the regions of interest in the (j, κ) -space shown in Figure 3.1. In each region c runs over the integers.

A first restriction of our study will be to focus on odd j that produce $Q_{j,\kappa,c}$ values involving G . Note that:

$$Q_{j,c,\kappa} = \begin{cases} \frac{a_0}{a_1 + a_2 G} & \text{if } j \text{ is odd} \\ \frac{a_0}{a_1 + a_2 \log 2} & \text{if } j \text{ is even} \end{cases}$$

Note as well that j, κ, c can be negative. In the examples given here we adopt the notation:

$$\frac{a_0}{a_1 + a_2 \cdot \text{constant}} = T(0) + \mathcal{K}_{n=1}^{\infty} \left(\frac{P(n)}{T(n)} \right)$$

This paper does not treat the even j case (a follow-up paper dealing with those cases is underway). Negative j, κ, c are of little interest as the Balkan’s numerator is defined as $-2n(c+n)(j+n-1)(1-j+2\kappa+n)$ hence, if $c < 0$ the term $(c+n)$ will hit zero for $n = -c$ and subsequently render the summation finite. The same happens for negative j with the term $(j+n-1)$ and for negative κ with the term $(1-j+2\kappa+n)$.

5.5 The Montenegro Conjecture

We start with the first j, κ space called Montenegro. Montenegro corresponds to the case $j = 1$. In other words:

$$Q_{1,\kappa,c} = 2\kappa + 1 + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n^2(n+2\kappa)(n+c)}{3n^2 + (3+4\kappa)n + 2\kappa + 1} \right)$$

Define the functions:

$$\Delta_{\kappa,c}(\alpha, \beta) = \begin{cases} \alpha + \beta c & \text{if } c < 2 \\ -2c(2c-1)(2(c-\kappa)-1)^2 \Delta_{\kappa,c-2}(\alpha, \beta) & \text{if } c \geq 2 \\ + (8c^2 + (2-8\kappa)c - 2\kappa + 1) \Delta_{\kappa,c-1}(\alpha, \beta) & \end{cases}$$

$$\Gamma_{\kappa,c}(\alpha, \beta) = (2c-1)!!^2 G + \Delta_{\kappa,c-1}(\alpha, \beta) \cdot \prod_{i=0}^{\kappa-1} (2(c-i)-1)$$

$$\delta_{\kappa} = \frac{4^{\kappa-1}}{(2\kappa-1)C_{\kappa-1}} \text{ and } \rho_{\kappa} = \frac{\delta_{\kappa}(-1)^{\kappa}(1-2\kappa)}{(2\kappa)!(2\kappa-3)!!}$$

$$\alpha_{\kappa} = \rho_{\kappa} \Delta_{1,\kappa-1}(1, -2) \text{ and } \beta_{\kappa} = -\rho_{\kappa} (2\kappa-3)^2 \Delta_{2,\kappa-1}(1, 12) - \alpha_{\kappa}$$

$$\text{Then } \forall \kappa, c \in \mathbb{N}^2, Q_{1,\kappa,c} = \frac{\delta_{\kappa}(2c)!}{\Gamma_{\kappa,c}(\alpha_{\kappa}, \beta_{\kappa})}$$

$Q_{1,\kappa,c}$ is hence an explicitly computable fraction of the form:

$$Q_{1,\kappa,c} = \frac{a_0}{a_1 + a_2 G} \text{ where } a_0, a_1, a_2 \in \mathbb{Z}$$

The code snippet testing this formula over the square $1 \leq \kappa, c \leq 14$ is entitled "1. Montenegro".

In summary in [Montenegro](#) we do not need to resort to any integer relation algorithms to compute $Q_{1,\kappa,c}$ for all κ, c values.

Remark 19. Note that, as described here, the complexity of $\Delta_{\kappa,c}$ is exponential in c , however, using classical Fibonacci memoization, this complexity can be reduced to $O(c \log c)$ thereby resulting in a very efficient algorithm for computing $Q_{1,\kappa,c}$.

5.6 The Bosnia & Herzegovina Conjecture

[Bosnia & Herzegovina](#) corresponds to the line $2\kappa = j - 3$, that is:

$$Q_{j, \frac{j-3}{2}, c} = -j + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n(n-2)(n+c)(n+j-1)}{3n^2 + (2j-3)n - j} \right) = 2 + 2c - j$$

We start by defining:

$$\alpha_j = 1 \text{ and } \beta_j = 15 - 4j$$

And let:

$$\Delta_{j,c}(\alpha_j, \beta_j) = \begin{cases} \alpha_j + \beta_j c & \text{if } c < 2 \\ -2c(2c-j)(2c+1)(2c-j+2) \Delta_{j,c-2}(\alpha_j, \beta_j) & \text{if } c \geq 2 \\ + (8c^2 + (14-4j)c - 3(j-2)) \Delta_{j,c-1}(\alpha_j, \beta_j) & \end{cases}$$

$$g(j, c) = \frac{(2c)!}{2} \prod_{i=1}^{\frac{j-1}{2}} (2+2i-j) \text{ and } h(j, c) = \prod_{i=1}^{\frac{j-5}{2}} (2c-2i-1)$$

Then:

$$Q_{j, \frac{j-3}{2}, c} = \frac{g(j, c)}{\Delta_{j, c-1}(\alpha_j, \beta_j) \cdot h(j, c)} \in \mathbb{Q}$$

Hence, in **Bosnia & Herzegovina** as well we do not need to resort to any integer relation algorithms to compute $Q_{j, \frac{j-3}{2}, c}$ which is an explicitly computable fraction in \mathbb{Q} .

The corresponding code snippet is "2. Bosnia".

Note that $Q_{j, \frac{j-3}{2}, c} = 2 + 2c - j$ gives a non-recursive formula for $\Delta_{j, c}(1, 15 - 4j)$.

This “long detour” for computing $Q_{j, (j-3)/2, c}$ which is a finite continued fraction (exactly computable by summation) is extremely useful as it unveiled the Δ structure and the basic ϕ_i s that repeatedly intervene in other Balkan areas.

5.7 Roadmap

We are now ready to describe the roadmap that will govern the rest of this paper.

5.7.1 Areas governed by one running variable

As we have just seen, **Montenegro** and **Bosnia & Herzegovina** follow similar behaviors. This pattern revolves around the “magic” values α, β which are formally known for **Montenegro** and **Bosnia & Herzegovina**. Both regions are lines parameterized by a single running variable (κ for **Montenegro** and j for **Bosnia & Herzegovina**).

5.7.2 The c -level master formula for all Balkans except **Montenegro**

Except **Montenegro** (whose case was settled) all other areas obey a common c -level master formula that we will provide below. Computing $Q_{j, \kappa, c}$ for any c using this c -level master formula requires knowledge of two rational parameters: $\alpha_{j, \kappa}$ and $\beta_{j, \kappa}$.

The cornerstone of the rest of this paper is thus the quest for $\alpha_{j, \kappa}$ and $\beta_{j, \kappa}$ for all Balkan areas except **Montenegro**.

$\alpha_{j, \kappa}, \beta_{j, \kappa}$ can always be inferred by computing numerically¹⁴ Q_{j, κ, c_1} and Q_{j, κ, c_2} for two values $c_1 \neq c_2$ and solving a system of two equations in the unknowns $\alpha_{j, \kappa}, \beta_{j, \kappa}$.

When $\alpha_{j, \kappa}, \beta_{j, \kappa}$ are found $Q_{j, \kappa, c}$ can be computed for any other $c \notin \{c_1, c_2\}$.

Using this process requires resorting twice to integer relation algorithms such as LLL [Lenstra, 1982], HJLS [Håstad, 1986], PSOS [Bailey, 1989; Ferguson, 1988] or PSLQ [Ferguson, 1999] for each (j, κ) -pair. This works perfectly in practice but is not entirely satisfactory because the process has a “blind spot” which is the integer relation oracle. We would like to avoid such blind spots as much as possible and dispose of a fully algebraic process for computing each $Q_{j, \kappa, c}$. The ideal situation being, of course, a direct algebraic computation of $Q_{j, \kappa, c}$ from the data j, κ, c alone.

¹⁴i.e. using an integer relation algorithm.

We completely achieve the goal of computing all $Q_{j,\kappa,c}$ values in an exact algebraic way all over the Balkans. Note that we do not need to bother with **Croatia** where $Q_{j,\kappa,c}$ is a finite continued fraction computable exactly by summation. This simply stems from the fact that as we increase n the $(1 - j + 2\kappa + n)$ component of Balkan's continued fraction will necessarily hit 0 in **Bosnia & Herzegovina** and in **Croatia**.

The snippet "3. Northern Balkans" illustrates the process with all areas except **Montenegro**. The snippet validates the c -level master formula on the 3D volume $3 \leq j \leq 13, 1 \leq \kappa \leq 6, 1 \leq c \leq 7$ (Figure 3.16) stretching over parts of **Croatia**, **Bosnia & Herzegovina**, **Serbia** and **Kosovo**. In each case the program derives the corresponding $\alpha_{j,\kappa}, \beta_{j,\kappa}$ and allows the computing of $Q_{j,\kappa,c}$ for any c . The formally computed results are then successfully compared to numerical ones.

As we will later see, for **Serbia** and **Kosovo** we have more powerful master formulae operating at the κ -level and the j -level.

The c -level master formula¹⁵ is defined as follows, assuming that we are somehow given the magic constants $\alpha_{j,\kappa}, \beta_{j,\kappa}$.

Let:

$$\Delta_{j,\kappa,c}(\alpha_{j,\kappa}, \beta_{j,\kappa}) = \begin{cases} \alpha_{j,\kappa} + \beta_{j,\kappa}c & \text{if } c < 2 \\ -2c(2c-j)(2c-2\kappa+j-2)(2c-2\kappa-1)\Delta_{j,\kappa,c-2}(\alpha_{j,\kappa}, \beta_{j,\kappa}) & \text{if } c \geq 2 \\ + (8c^2 + (2-8\kappa)c + (j-2)(2\kappa-j))\Delta_{j,\kappa,c-1}(\alpha_{j,\kappa}, \beta_{j,\kappa}) & \end{cases}$$

$$f_{j,\kappa,c} = C_{\frac{j-3}{2}} C_{\kappa-1} (j-2)(2\kappa-1)(2c-1)!!^2 \prod_{i=1}^{\frac{j-1}{2}} (2c-2\kappa+2i-1)(\kappa-i+1)$$

$$g_{j,\kappa,c} = (2c)! 2^{\frac{j+4\kappa-7}{2}} \prod_{i=1}^{\frac{j-1}{2}} (2c-2i+1)(2\kappa-2i+1)$$

$$h_{j,\kappa,c} = \prod_{i=0}^{\frac{j-3}{2}} (2c-2i-1) \prod_{i=0}^{\kappa-1} (2c-2i-1)$$

Then:

$$Q_{j,\kappa,c} = \frac{g(j, \kappa, c)}{\Delta_{j,\kappa,c-1}(\alpha_{j,\kappa}, \beta_{j,\kappa}) \cdot h(j, \kappa, c) + f(j, \kappa, c) \cdot G}$$

Remark 20. For **Bosnia & Herzegovina** and **Croatia** $f_{j,\kappa,c} = 0$. This happens automatically given the definition of $f_{j,\kappa,c}$.

5.8 The **Serbia** conjecture

Montenegro, **Kosovo** and **Serbia** feature internal symmetries illustrated by the arches in Figures 3.1 where connected Q values are identical. We will also provide a formula connecting their α, β values over the plan. This means that any formula applicable to **Kosovo** will also settle the case of **Serbia** (given that **Montenegro** is settled

¹⁵valid for all areas except **Montenegro** whose case was anyhow previously settled.

and given that **Montenegro** provides the values for the upper border line of **Serbia**).

Because:

$$Q_{1+2i,\kappa,c} = Q_{2\kappa+1-2i,\kappa,c} \text{ valid for } \kappa = 2, 3, \dots \text{ and } 0 \leq i \leq \left\lfloor \frac{\kappa}{2} \right\rfloor - 1$$

To compute any value in **Serbia**, we can just compute its symmetric correspondent in **Kosovo**.

5.9 The **Kosovo** Conjecture

Kosovo is very specific in that it has a κ -level master formula. This means that for each j , knowledge of four upper-level rational constants¹⁶:

$$\{\bar{\alpha}_j, \bar{\beta}_j\} = \left\{ \{\alpha_j^\alpha, \alpha_j^\beta\}, \{\beta_j^\alpha, \beta_j^\beta\} \right\}$$

allows to generate $\alpha_{j,\kappa}, \beta_{j,\kappa}$ for all κ values along a j -line within **Kosovo**.

Finding $\bar{\alpha}_j, \bar{\beta}_j$ for a given j requires solving a system of equations using four known $\alpha_{j,\kappa_1}, \beta_{j,\kappa_1}, \alpha_{j,\kappa_2}, \beta_{j,\kappa_2}$ for some $\kappa_1 \neq \kappa_2$. To determine the two $\alpha_{j,\kappa_i}, \beta_{j,\kappa_i}$ pairs¹⁷ we can solve two systems of equations (each in two unknowns) using any $Q_{j,\kappa_1,c_1}, Q_{j,\kappa_1,c_2}, Q_{j,\kappa_2,c_3}$ and Q_{j,κ_2,c_4} . Note that there is no opposition to take $c_1 = c_3$ and $c_2 = c_4$. This process is illustrated in Figure 3.5.

5.9.1 The **Kosovo** κ -level master formula

The κ -level master formula for **Kosovo** allows to infer $\alpha_{j,\kappa}, \beta_{j,\kappa}$ from $\bar{\alpha}_j, \bar{\beta}_j$ for a fixed j and a variable κ .

In other words, the **Kosovo** κ -level master formula generates for a fixed j and for a variable κ the data $\alpha_{j,\kappa}, \beta_{j,\kappa}$ necessary to operate the general c -level master formula given in subsection 5.7.2.

Recall that the c -level master formula of subsection 5.7.2 generates for a fixed j, κ and a variable c a formal expression of the continued fraction $Q_{j,\kappa,c}$, given the auxiliary input $\alpha_{j,\kappa}, \beta_{j,\kappa}$.

Define:

$$\pi(j, \kappa) = \prod_{i=0}^{\frac{j-3}{2}} (\kappa - i)(2\kappa - 2i - 1)^2$$

$$\ell(n, j, \kappa) = \frac{(-1)^{\kappa+1} (2\kappa)!^2}{\kappa! 2^{3\kappa-2} (2\kappa - j)(2\kappa - 1)(n((2\kappa - j - 2)(3 - 2\kappa) - 1) + 1) \cdot \pi(j, \kappa)}$$

$$\eta(n, j, \kappa) = (2\kappa + 2j - 9 - 2n)(2\kappa + j - 8 - 2n)(-2\kappa + 5 - j)(2\kappa + j - 6)$$

$$\phi(n, j, \kappa) = 8\kappa^2 + \kappa(10j - 48 - 8n) + 3j^2 - (28 + 4n)j + 68 + 18n$$

¹⁶This somewhat unusual notation is used to note the “ α of α ”, the “ β of α ” etc., as the same type of formula is applied at both c and κ levels.

¹⁷for $i \in \{1, 2\}$.

$$\bar{\Delta}_{n,j,\kappa}(\alpha, \beta) = \begin{cases} \alpha + \beta\kappa & \text{if } \kappa < 2 \\ \eta(n, j, \kappa) \cdot \bar{\Delta}_{n,j,\kappa-2}(\alpha, \beta) + \phi(n, j, \kappa) \cdot \bar{\Delta}_{n,j,\kappa-1}(\alpha, \beta) & \text{if } \kappa \geq 2 \end{cases}$$

We assume that we are given the four constants:

$$\{\bar{\alpha}_j, \bar{\beta}_j\} = \left\{ \left\{ \alpha_j^\alpha, \alpha_j^\beta \right\}, \left\{ \beta_j^\alpha, \beta_j^\beta \right\} \right\} \in \mathbb{Q}^4$$

Then:

$$\alpha_{j,\kappa} = \frac{\bar{\Delta}_{0,j,\kappa-j+2}(\alpha_j^\alpha, \alpha_j^\beta)}{\ell(0, j, \kappa)} \quad \text{and} \quad \beta_{j,\kappa} = \frac{\bar{\Delta}_{1,j,\kappa-j+2}(\beta_j^\alpha, \beta_j^\beta)}{\ell(1, j, \kappa)} - \alpha_{j,\kappa}$$

The process is illustrated by the code snippet "4. Kosovo".

The computation of the "magic" lists of constants $\{\bar{\alpha}_j, \bar{\beta}_j\}$ hard-coded in the snippet "4. Kosovo" is done by resorting to integer relation resolution in snippet "5. resolution". A formal computation of $\bar{\alpha}_{j+2}, \bar{\beta}_{j+2}$ from $\bar{\alpha}_j, \bar{\beta}_j$ is given in the next subsection.

Remark 21. It is interesting to note that thanks to the inner symmetry within [Kosovo](#), it is possible to determine $\bar{\alpha}_{j+2}, \bar{\beta}_{j+2}$ if $\bar{\alpha}_j, \bar{\beta}_j, \bar{\alpha}_{j+4}, \bar{\beta}_{j+4}$ are known. Taking as an example $j = 5$, Figure 3.1 shows that $Q_{5,3,c} = Q_{3,3,c}$. Hence knowledge of $\bar{\alpha}_3, \bar{\beta}_3$ will be used to compute $Q_{3,3,c}$ which is identical to $Q_{5,3,c}$.

Note that $Q_{5,5,c} = Q_{7,5,c}$. Hence knowledge of $\bar{\alpha}_7, \bar{\beta}_7$ will be used to compute $Q_{7,5,c}$ which is identical to $Q_{5,5,c}$. Finally, having in hand $Q_{5,3,c}, Q_{5,5,c}$ we can solve a system in two unknowns and determine $\bar{\alpha}_5, \bar{\beta}_5$. Unfortunately this process has an information flow that only operates in "sandwich mode" allowing to determine $\bar{\alpha}_{j+2}, \bar{\beta}_{j+2}$ from $\bar{\alpha}_j, \bar{\beta}_j, \bar{\alpha}_{j+4}, \bar{\beta}_{j+4}$. This information flow cannot be reversed into an "escalator" allowing to ascend to level $j+4$ from levels j and $j+2$. The situation is illustrated in Figure 3.6 where $x \rightarrow y$ denotes the relation "y is computable from x". This limitation is solved at the next subsection where we infer $\bar{\alpha}_{j+2}, \bar{\beta}_{j+2}$ from $\bar{\alpha}_j, \bar{\beta}_j$.

5.9.2 The [Kosovo](#) j -level master formula

Define:

$$\varrho(j) = 2^j \left(\frac{j-1}{2}! \right)^2 C_{\frac{j-3}{2}}$$

$$\alpha_{j+2}^\alpha = \frac{4 \left(\alpha_j^\alpha (j-4)(j-1)j(2j-5)(2j-3) - (3j-2)\varrho(j) \right)}{(j-4)(j-1)(j+1)}$$

The formula is valid from $j = 3$ and on (for which $\alpha_3^\alpha = -1$).

$$\alpha_{j+2}^\beta = \frac{4 \left(\alpha_j^\beta j(2j+1)(j-1)(2j-5)(j-4) - (3j^3 - 17j^2 + 11j + 18)\varrho(j) \right)}{(j-4)(j-3)(j+1)}$$

The formula is valid from $j = 5$ and on (for which $\alpha_5^\beta = 234$).
 For $j = 3$ use directly $\alpha_3^\beta = 4$.

$$\alpha_{j+2}^\beta = \frac{4 \left(j(j-6)(j-4)(j-2)(j-1)(2j-7)(2j-5)\beta_j^\alpha - 12(j^2-4j+2)\varrho(j) \right)}{(j-6)(j-4)^2(j-1)(j+1)}$$

The formula is valid from $j = 3$ and on (for which $\beta_3^\alpha = -1/3$).

The formula for β_{j+2}^β is much more complex and does not fit in a single line.

Define:

$$\vartheta_1(j) = (j-6)(j-4)(j-2)(j-1)j(2j-7)(2j-5)(-1-3j+2j^2)$$

$$\vartheta_2(j) = 4(-390 + 312j + 653j^2 - 942j^3 + 442j^4 - 87j^5 + 6j^6)$$

$$\vartheta_3(j) = (j-6)(j-4)^2(j-1)(1+j)(13-11j+2j^2)$$

Then:

$$\beta_{j+2}^\beta = \frac{4 \left(\beta_j^\beta \vartheta_1(j) - \vartheta_2(j)\varrho(j) \right)}{\vartheta_3(j)}$$

The formula is valid from $j = 3$ and on (for which $\beta_3^\beta = -14/3$).

The test of those formulae proceeds in two steps. A first snippet ("16. files") computes the $\bar{\alpha}_j$ and $\bar{\beta}_j$ for $j = 3, 5, \dots, 501$ using numerical simulation and records them in two files called `file1.txt` and `file2.txt`. Those files are read by snippet "17. j-level" that compares them to the values derived using the formal j -level formulae.

Remark 22. It is also possible to directly infer the $\alpha_{j,\kappa}, \beta_{j,\kappa}$ through symmetry. Note that the relation below also works for negative j, κ values.

Define:

$$\zeta(j, u) = \frac{1}{2^u} \prod_{i=0}^{u-1} (2+2i-j)$$

$$\tau_{j,u} = \begin{cases} \frac{\zeta(j, u)}{|\zeta(j, u)|} \cdot (2j-2u-3)!!(2u-j)!!(-2)^u & \text{if } 2u > j-1 \\ \zeta(j, u) \cdot (-4)^u & \text{otherwise} \end{cases}$$

i	$\psi_1(i, j)$
0	-1
1	$14 - j$
2	$-464 + 58j - 3j^2$
3	$27936 - 4692j + 432j^2 - 15j^3$
4	$-2659968 + 542256j - 67836j^2 + 4260j^3 - 105j^4$
5	$367568640 - 86278560j + 13203480j^2 - 1139700j^3 + 51450j^4 - 945j^5$

Table 3.15: $\psi_1(i, j)$ for $0 \leq i \leq 5$.

Then:

$$\frac{\alpha_{j, j-u-1}}{\alpha_{j-2u, j-u-1}} = \frac{\beta_{j, j-u-1}}{\beta_{j-2u, j-u-1}} = \tau_{j, u}$$

In particular $\tau_{j, j-1} = \frac{1}{2j-4}$. The code is snippet "6. symmetry".

5.10 The Croatia Conjecture

As [Bosnia & Herzegovina](#) is a particular border case of [Croatia](#) the following is valid for both [Bosnia & Herzegovina](#) and [Croatia](#).

Our automated software detected the following **stunning** behavior providing a κ -level formula for [Croatia](#).

Let:

$$\mu_{i, j} = -(-2)^{\frac{3j-11-4i}{2}} \prod_{q=1}^i (j - 2q - 2)$$

For every $i = 0, 1, 2, \dots$ there exist two polynomials in j , denoted $\psi_1(i, j)$ and $\psi_2(i, j)$ such that for $j \geq 2i + 5$ we have:

$$\alpha_{j, \frac{j-2i-3}{2}} = \frac{\psi_1(i, j)}{\mu(i, j)} \quad \text{and} \quad \beta_{j, \frac{j-2i-3}{2}} = \frac{\psi_2(i, j)}{\mu(i, j)}$$

Tables 3.15 and 3.16 provide the first values of the polynomials ψ_1, ψ_2 .

ψ_1, ψ_2 can be computed algebraically because when $(j, \kappa, c) \in \text{Croatia}$, $Q_{j, \kappa, c}$ can be obtained by finite summation. We can hence derive $\alpha_{j, \frac{j-2i-3}{2}}$ and $\beta_{j, \frac{j-2i-3}{2}}$.

Then, by deriving enough $(\alpha_{j, \bullet}, \beta_{j, \bullet})$ pairs and knowing that $\deg_j \psi_1(i, j) = i$ and $\deg_j \psi_2(i, j) = i + 1$, we can compute $\psi_1(i, j)$ and $\psi_2(i, j)$ by interpolation. We did not code this tedious yet straightforward process.

See code snippet "7. Croatia".

Remark 23.

The leading coefficients of $\psi_1(i, j)$ (i.e. 1, 1, 3, 15, 105, 945, ...) are $(2i - 1)!!$ whereas the leading coefficients of $\psi_2(i, j)$ (i.e. 4, 4, 12, 60, 420, 3780, ...) are $4(2i - 1)!!$.

Remark 24. The ψ polynomials can always be written under a nested form, e.g.:

i	$\psi_2(i, j)$
0	$-15 + 4j$
1	$306 - 95j + 4j^2$
2	$-13360 + 4646j - 357j^2 + 12j^3$
3	$999648 - 379692j + 40368j^2 - 2457j^3 + 60j^4$
4	$-113885568 + 46449360j - 6124164j^2 + 513228j^3 - 22935j^4 + 420j^5$
5	$18333538560 - 7933530720j + 1224286440j^2 - 126833100j^3 + 7864950j^4 - 266175j^5 + 3780j^6$

Table 3.16: $\psi_2(i, j)$ for $0 \leq i \leq 5$.

$$\begin{aligned} \psi_1(j, 6) = & -14487726825 - (104826150 + (452605725 \\ & + (121200300 + (13697775 + (640710 \\ & + 10395 \cdot (j - 27))(j - 25))(j - 23))(j - 21))(j - 19))(j - 17) \end{aligned}$$

$$\begin{aligned} \psi_2(j, 6) = & 3198013886925 + (145296572850 + (5207427225 \\ & + (4353102000 + (877052475 + (78210090 + (3023055 \\ & + 41580 \cdot (j - 29))(j - 27))(j - 25))(j - 23))(j - 21))(j - 19))(j - 17) \end{aligned}$$

Remark 25. The GCD between the u -th coefficient of $\psi_1(i, j)$ and the u -th coefficient of $\psi_2(i, j)$ is always smooth as illustrated in the code snippet "8. coefficients".

5.11 Balkans Knowledge Summary

In summary:

- Any $Q_{j,\kappa,c}$ value in **Bosnia & Herzegovina** is algebraically computable either by summation or by the **Bosnian** κ -level formula.
- In **Croatia** $Q_{j,\kappa,c}$ is directly computable by summation (hence making a j -level formula superfluous) and, in addition, has κ -level formulae.
- Any $Q_{j,\kappa,c}$ value in **Montenegro** is algebraically computable.
- For **Kosovo** we have j -level formulae.
- **Serbia** is fully determined by our knowledge of **Montenegro** and **Kosovo**.
- Negative j, κ, c values are of no interest as they provide finite summations.

We therefore have algebraic formulae for computing all $Q_{j,\kappa,c}$ values over all the Balkans.

A challenge, on which the authors are currently working, is characterizing the case of even j values (that yield formulae involving $\log 2$).

As a motivational example let us unveil the simple $\log 2$ example $Q_{2,\kappa,0}$.

In this case:

$$Q_{2,\kappa,0} = \frac{(-1)^{\kappa+1} a_\kappa}{-b_\kappa + a_\kappa \log 2}$$

Area	j -level formula	κ -level formula	c -level formula
Croatia	not required	✓	✓
Bosnia & Herzegovina	not required	✓	✓
Kosovo+Serbia	✓	✓	✓
Montenegro	not required	✓	✓

Table 3.17: Knowledge Summary

Where a_κ is the LCM of the list of κ integers starting with κ (in Mathematica: `Table[Apply[LCM, Table[i, {i, k, 2k-1}]], {k, 1, 100}]`) while the b_κ are the numerators of the coefficients in the power series for $-\log(1+x)\log(1-x)$.

Recall that the coefficients in the expansion of $\log(1+x)\log(1-x)$ are given by

$$\frac{1}{2} \binom{2n}{n} \int_{x=0}^1 (x(1-x))^{n-1} \log(x) dx$$

and

$$\log(1+x)\log(1-x) = \frac{1}{2} \int_{z=0}^1 \frac{\log(z)}{z(1-z)} \left(\frac{1}{\sqrt{1-4x^2z(1-z)}} - 1 \right) dz$$

Evidently, providing formal proofs of the formulae provided in this paper is a challenge by its own right. The Conservative Matrix Field approach of [David, 2023] is one promising direction to investigate. To date, attempts to code automated substitution-simplification-induction proofs were unsuccessful. Yet another interesting question is that of *reversal*: Given a $Q_{j,\kappa,c}$ (i.e. a_0, a_1, a_2) find j, κ, c ¹⁸.

The main open questions remaining are very simple to formulate:

Open Question 1

Prove the formulae given in this paper.

Open Question 2

What happens in the Balkans for even j values?

Open Question 3

What happens in Inostranstvo (cf. remark 30)?

The following observations are hints that may serve in future quests:

Remark 26.

$$Q_{1,0,c} = \frac{(2c)!}{2(2c-1)!!^2 G - \Delta_{c-1,0}}$$

where:

¹⁸Reversal is not always possible over Montenegro because $\forall \kappa, Q_{1,\kappa,1} = Q_{1,1,\kappa}$.

$$\Delta_{1,0,c} = \begin{cases} 1 + 10c & \text{if } c < 2 \\ 2c(1-2c)^3 \Delta_{1,0,c-2} + (8c^2 + 2c + 1) \Delta_{1,0,c-1} & \text{if } c \geq 2 \end{cases}$$

or under an equivalent more compact form:

$$\Delta_{1,0,c} = \begin{cases} 1 & \text{if } c = 0 \\ (2c)! + (2c+1)^2 \Delta_{1,0,c-1} & \text{if } c > 0 \end{cases}$$

and even [Cloître, 2004]:

$$\Delta_{1,0,c-1} = (2c)! \left(\frac{2G\left(\frac{2c}{c}\right)}{4^c} - \int_0^\infty \frac{t}{\cosh^{2c+1}(t)} dt \right)$$

Remark 27.

$$\lim_{c \rightarrow \infty} Q_{j,\kappa,c+1} - Q_{j,\kappa,c} = 2$$

$$\lim_{\kappa \rightarrow \infty} Q_{j,\kappa+1,c} - Q_{j,\kappa,c} = 2j$$

$$\lim_{\kappa \rightarrow \infty} Q_{j+1,j+2r+1,c} - Q_{j,j+2r+1,c} = 4r + 1$$

Remark 28. Denoting:

$$Q_{j,\kappa,c} = \frac{a_0}{a_1 + a_2 G} \text{ where } a_0, a_1, a_2 \in \mathbb{Z}$$

The following formula (code snippet "9. ratio") is valid all over areas:

$$\rho_{j,\kappa,c} = \prod_{i=1}^{\frac{j-1}{2}} \frac{(2c-2\kappa+2i-1)(\kappa-i+1)}{(2c-2i+1)(2\kappa-2i+1)} \text{ and } \varepsilon_{j,\kappa} = 2\kappa + \frac{j-7}{2} + \left\lfloor \frac{1}{j} \right\rfloor$$

$$\frac{a_0}{a_2} = \frac{(2c)! \cdot 2^{\varepsilon_{j,\kappa}}}{(2c-1)!!^2 \cdot C_{\kappa-1} \cdot C_{\frac{j-3}{2}} \cdot (2\kappa-1) \cdot (j-2) \cdot \rho_{j,\kappa,c}}$$

Where $\rho_{1,\kappa,c} = 1$ by definition.

Remark 29. Although possibly unrelated, we note that low-degree continued fractions involving $\log 2$ can be also obtained with lower degree polynomials, e.g. (See code snippet "10. log2-a"):

$$\frac{2}{L\left(\frac{1}{2}, 1, c-1\right)} = \frac{2}{\sum_{n=0}^{\infty} \frac{e^{\pi i n}}{(n+c-1)}} = \frac{1}{2^{c-2} \log(2) - \sum_{j=1}^{c-2} \frac{2^{c-j-2}}{j}} = c + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n^2}{3n+c} \right)$$

We get a similar behavior for:

$$R_c = c + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n^2 - 2n}{3n+c} \right)$$

where $2^{c-4} \cdot (c-3) \cdot a_0 = a_2$ and for which we provide numerical examples in Table 3.25.

See code snippet "11. log2-b".

Remark 30. We also discovered other continued fractions involving G outside the Balkans. This suggests the existence of a more general formula encompassing both the Balkans and those other territories (called “Inostranstvo”).

We denote those relations:

$$Q'_{\delta,\epsilon,\tau,\eta,\mu} = \epsilon + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n(n+\tau)(n+\eta)(n+\mu)}{\epsilon + \delta n + 3n^2} \right) = \frac{a_0}{a_1 + a_2 G} \text{ where } a_0, a_1, a_2 \in \mathbb{Z}$$

Luckily, given that we have two coefficients¹⁹ in the denominator of the continued fractions, we can compare in one plot the coefficients ϵ and δ of the Balkans and of Inostranstvo in one figure (Figure 3.9). Similarly we can visualize in 3D the τ, η, μ of both regions (Figure 3.10). The alignments of red points show that there is clearly another structured family hiding out beyond the Balkans.

PCA and automated matching revealed that ϵ, δ are dependent on τ, η, μ and:

$$Q'_{\tau,\eta,\mu} = x + 2(\tau + \eta + 1)i + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n(n+\tau)(n+\eta)(n+2i+\mu)}{x + 2(\tau + \eta + 1)i + (2(\tau + \eta + \mu + 2i) + 3)n + 3n^2} \right)$$

Where $x = (1 + \eta)(1 + \mu) + \tau(1 + \eta + \mu)$.

This formula works when all variables²⁰ have identical parity, i.e.:

$$\tau \equiv \eta \equiv \mu \pmod{2}$$

For instance:

$$Q'_i = 7 + 6i + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n(n+1)^2(n+2i+1)}{7 + 6i + (9 + 4i)n + 3n^2} \right) \text{ for } i = 0, 1, \dots$$

Whose formal expression (code snippet "12. Inostranstvo1") turns out to be:

$$Q'_i = \frac{(2i+1)!}{\Delta'_i - 2G(2i+1)!!^2}$$

$$\Delta'_i = \begin{cases} 2 + 15i & \text{if } i < 2 \\ 2(2i-1)^3(1-i) \cdot \Delta'_{i-2} + (8i^2 - 2i + 3) \cdot \Delta'_{i-1} & \text{if } i \geq 2 \end{cases}$$

or²¹:

$$Q''_i = 23 + 10i + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n(n+1)(n+3)(n+2i+3)}{23 + 10i + (17 + 4i)n + 3n^2} \right) \text{ for } i = 0, 1, \dots$$

¹⁹We exclude the 3 of $3n^2$ in the denominator, which is common to both the Balkans and to Inostranstvo.

²⁰(code snippet "14. Inostranstvo")

²¹for which

$$\frac{a_0}{a_2} = \frac{(2j+5)!}{(2j+4)(2j+5)!!^2}$$

(code snippet "13. Inostranstvo2")

We did not investigate further the various Inostranstvo families but conjecture that they share the same behaviors as the Balkans.

Remark 31. The Δ functions appearing in this paper are particular cases of “Generalized Fibonacci Polynomials” (GFBs) studied by various authors, e.g. [Flórez, 2019]. GFBs have numerous properties that might shed light on the open questions listed *supra*. We did not investigate this further.

Yet another route consists in considering the Inostranstvo and/or even- j targets as algebraic equations and attempting on them an algebraic sieving approach such as [Barral, 2021]. This was not investigated this process given its theoretical and logistical complexity.

Finally, the similarity between the infinite sums given in [Nimbran, 2018] and the continued fractions investigated in this paper may reveal connections allowing to prove our conjectures. While analyzing [Nimbran, 2018] we noted a probable misprint in the formulae given for y_3 and y_5 (bottom of page 9 of [Nimbran, 2018]). We hence conducted our own experiments and discovered the relations described in Table 3.18 where:

$$w_2c + w_3 + w_1 \sum_{n=1}^{\infty} (-1)^{n+1} \prod_{i=1}^7 (2n + 2i - 3 + \epsilon)^{-e_i} = 0$$

5.12 The Gradient Descent Process

The Gradient Descent process that generated the formulae will be presented in steps. We sample execution at critical points using the example $j = 11, \kappa = 6$ and $40 \leq c \leq 47$ to explain the automated exploration process.

5.12.1 The starting point

We first recall our notations:

$$Q_{j,\kappa,c} = \frac{a_0}{a_1 + a_2 G} \text{ where } a_0, a_1, a_2 \in \mathbb{Z}$$

We have by now “seen the end of the movie”, and we know that for all areas:

$$n_0(j, \kappa, c) = \frac{a_0}{a_2} = \frac{(2c)! \cdot 2^{\varepsilon_{j,\kappa}}}{(2c-1)!!^2 \cdot C_{\kappa-1} \cdot C_{\frac{j-3}{2}} \cdot (2\kappa-1) \cdot (j-2) \cdot \rho_{j,\kappa,c}}$$

Where:

$$\rho_{j,\kappa,c} = \prod_{i=1}^{\frac{j-1}{2}} \frac{(2c-2\kappa+2i-1)(\kappa-i+1)}{(2c-2i+1)(2\kappa-2i+1)} \text{ and } \varepsilon_{j,\kappa} = 2\kappa + \frac{j-7}{2} + \left\lfloor \frac{1}{j} \right\rfloor$$

We started our journey by manually inspecting $n_0(j, \kappa, c)$ for several j, κ, c values, noting that $n_0(j, \kappa, c)$ is always very smooth.

This suggests that $n_0(j, \kappa, c)$ is the product of basic combinatorial functions such as factorials, binomials, semifactorials, Catalan numbers, Pochhammer symbols etc.

But the question is – of course – **which functions?**

We now know that $n_0(j, \kappa, c)$ is an exotic zoo containing the following animals:

$$\phi_0 = (2c)!, \quad \phi_1 = (2c-1)!!, \quad \phi_2 = (2c-1)!!, \quad \phi_3 = C_{\kappa-1}, \quad \phi_4 = C_{\frac{j-3}{2}}$$

ϵ	c	e_1, e_2, \dots, e_7	w_1, w_2, w_3
0	π	0, 0, 2, 2, 2, 2, 0	57153600, -33075, 103904
0	π	0, 1, 1, 2, 2, 1, 1	69854400, -24255, 76192
0	π	0, 1, 2, 1, 1, 2, 1	62868960, -14553, 45712
0	π	0, 2, 2, 1, 2, 2, 0	-65318400, -4725, 14848
0	π	0, 2, 2, 2, 2, 0, 0	-6350400, -3675, 11552
0	π	1, 1, 2, 2, 1, 1, 0	-907200, -315, 992
0	π	1, 2, 1, 1, 2, 1, 0	-635040, -147, 464
0	π	1, 2, 2, 2, 2, 1, 0	16934400, -245, 768
0	π	2, 1, 1, 1, 1, 2, 0	-59535000, -6615, 21292
0	π	2, 2, 0, 0, 2, 1, 1	-93139200, -2695, 9344
0	π	2, 2, 0, 0, 2, 2, 0	-52920000, -2695, 9056
0	π	2, 2, 1, 2, 2, 0, 0	-50803200, 3675, -11264
0	π	2, 2, 2, 2, 0, 0, 0	129600, -75, 224
0	G	0, 1, 2, 2, 2, 1, 0	-50803200, 66150, -60577
0	G	0, 3, 2, 3, 0, 0, 0	-3456000, -6750, 6197
0	G	1, 2, 2, 2, 1, 0, 0	-2419200, -3150, 2909
0	G	1, 3, 0, 3, 1, 0, 0	8064000, 8750, -8109
0	G	2, 2, 0, 2, 2, 0, 0	-33868800, -22050, 21131
0	G	3, 2, 3, 0, 0, 0, 0	-27648, 54, -25
1	$\log 2$	0, 0, 0, 2, 2, 2, 2	-33177600, -38400, 26617
1	$\log 2$	0, 0, 2, 2, 2, 2, 0	1382400, -1600, 1109
1	$\log 2$	0, 1, 1, 2, 2, 1, 1	11059200, -7680, 5323
1	$\log 2$	0, 1, 2, 1, 1, 2, 1	-22118400, 10240, -7097
1	$\log 2$	0, 2, 2, 0, 0, 2, 2	17280000, -1760, 1219
1	$\log 2$	0, 2, 2, 2, 2, 0, 0	442368, 512, -355
1	$\log 2$	1, 1, 2, 0, 0, 2, 2	-88473600, 5120, -3539
1	$\log 2$	1, 1, 2, 2, 1, 1, 0	-230400, -160, 111
1	$\log 2$	1, 2, 1, 1, 2, 1, 0	-22118400, -10240, 7109
1	$\log 2$	2, 2, 0, 0, 2, 1, 1	-88473600, -5120, 3627
1	$\log 2$	2, 2, 0, 0, 2, 2, 0	69120000, 7040, -4951
1	$\log 2$	2, 2, 1, 2, 2, 0, 0	7077888, -1024, 707
1	$\log 2$	2, 2, 2, 2, 0, 0, 0	-13824, 16, -11

Table 3.18: Relations for π , G and $\log 2$. See code snippet "15. Series".

$$\begin{aligned}
\phi_5 &= (2\kappa - 1), \quad \phi_6 = (j - 2), \quad \phi_7 = 2^{2\kappa}, \quad \phi_8 = 2^{\lfloor \frac{j-7}{2} \rfloor}, \quad \phi_9 = 2^{\lfloor \frac{j}{2} \rfloor} \\
\phi_{10} &= \prod_{i=1}^{\lfloor \frac{j-1}{2} \rfloor} (2c - 2\kappa + 2i - 1), \quad \phi_{11} = \prod_{i=1}^{\lfloor \frac{j-1}{2} \rfloor} (\kappa - i + 1), \quad \phi_{12} = \prod_{i=1}^{\lfloor \frac{j-1}{2} \rfloor} (2c - 2i + 1) \\
\phi_{13} &= \prod_{i=1}^{\lfloor \frac{j-1}{2} \rfloor} (2\kappa - 2i + 1) \\
n_0(j, \kappa, c) &= \frac{\phi_0 \cdot \phi_7 \cdot \phi_8 \cdot \phi_9 \cdot \phi_{12} \cdot \phi_{13}}{\phi_1 \cdot \phi_2 \cdot \phi_3 \cdot \phi_4 \cdot \phi_5 \cdot \phi_6 \cdot \phi_{10} \cdot \phi_{11}}
\end{aligned}$$

However, at start, we had no idea what the ϕ_i s were nor do we know how many ϕ_i s are there.

Remark 32. The basic functions in our catalog are not independent as some multiplicatively generate others. e.g., Catalan numbers, binomials, multinomials and Pochhammer symbols are all products of factorials. Adding to the catalog 2^x we reach semifactorials etc. The code can hence *successfully* follow different paths for a given target $n_0(j, \kappa, c)$. While those functional dependencies do not impact the final result, they do impact complexity: e.g., using Catalan numbers *reduces the depth of search*²² but *increases its width*.

Conversely, the code can also remove successfully components of ϕ_i s and get stuck later if the remaining (un-removed part) of the ϕ_i is absent from the catalog.

5.12.2 Mutating functions

The algorithm performs a gradient descent on $n_i(j, \kappa, c)$, using an LLM to guide the descent. Catalog functions are not used in “bare metal” mode. They appear with specific linear combinations of $j, \kappa, c, 1$. To capture those combinations let:

$$\sigma(\bar{u}) = u_0 j + u_1 \kappa + u_2 c + u_3 \quad \text{where } \bar{u} = \{u_0, u_1, u_2, u_3\}$$

In other words²³:

$$\begin{aligned}
\phi_0 &= \sigma(0, 0, 2, 0)!, \quad \phi_1 = \phi_2 = \sigma(0, 0, 2, -1)!!, \quad \phi_3 = C_{\sigma(0, 1, 0, -1)} \\
\phi_4 &= C_{\sigma(\frac{1}{2}, 0, 0, -\frac{3}{2})}, \quad \phi_5 = \sigma(0, 2, 0, -1), \quad \phi_6 = \sigma(1, 0, 0, -2), \quad \phi_7 = 2^{\sigma(0, 2, 0, 0)} \\
\phi_8 &= 2^{\sigma(\frac{1}{2}, 0, 0, -\frac{7}{2})}, \quad \phi_{10} = \prod_{i=1}^{\sigma(\frac{1}{2}, 0, 0, -\frac{1}{2})} (\sigma(0, -2, 2, -1) + 2i) \\
\phi_{11} &= \prod_{i=1}^{\sigma(\frac{1}{2}, 0, 0, -\frac{1}{2})} (\sigma(0, 1, 0, 1) - i), \quad \phi_{12} = \prod_{i=1}^{\sigma(\frac{1}{2}, 0, 0, -\frac{1}{2})} (\sigma(2, 0, 0, 1) - 2i) \\
\phi_{13} &= \prod_{i=1}^{\sigma(\frac{1}{2}, 0, 0, -\frac{1}{2})} (\sigma(0, 2, 0, 1) - 2i)
\end{aligned}$$

²²When a Catalan number is identified three factorial identifications are avoided.

²³ $\phi_9 = 2^{\lfloor \frac{j}{2} \rfloor}$ was not in the catalog and was added manually to unify two families of general formulae discovered by two runs.

5.12.3 What information do we have?

The integer relations oracle²⁴ provides tens of thousands of $n_0(j, \kappa, c)$ instances. We record these in a database B where each entry has the form:

$$B_i = \{j_i, \kappa_i, c_i, t_i\} = \{j_i, \kappa_i, c_i, n_0(j_i, \kappa_i, c_i)\}$$

The code needs to infer the \bar{u} s that intervene in $n_0(j, \kappa, c)$ from those numerous numerical examples.

5.12.4 A first step

Assume that the code is discovering the ϕ_i s and their \bar{u} s one by one. The code is at some step ω at which it has already discovered the ϕ_i s shown in red in the formulae:

$$n_0(j, \kappa, c) = \frac{a_0}{a_2} = \frac{(2c)! \cdot 2^{\varepsilon_{j,\kappa}}}{(2c-1)!!^2 \cdot C_{\kappa-1} \cdot C_{\frac{j-3}{2}} \cdot (2\kappa-1) \cdot (j-2) \cdot \rho_{j,\kappa,c}}$$

And:

$$\rho_{j,\kappa,c} = \frac{\prod_{i=1}^{\frac{j-1}{2}} (2c-2\kappa+2i-1)(\kappa-i+1)}{\prod_{i=1}^{\frac{j-1}{2}} (2c-2i+1)(2\kappa-2i+1)}$$

Divide the t_i of each record B_i by the red components evaluated at j_i, κ_i, c_i . Update the target to:

$$n_\omega(j, \kappa, c) = (2c-1)!!^2 \cdot C_{\kappa-1} \cdot C_{\frac{j-3}{2}} \cdot (2\kappa-1) \cdot (j-2) \cdot \prod_{i=1}^{\frac{j-1}{2}} (2c-2\kappa+2i-1)(\kappa-i+1)$$

A successfully mutated function, e.g. $\phi_1 = (2c-1)!! = \sigma(0, 0, 2, -1)!!$ stands-out because:

$$n_\omega(j, \kappa, c) \bmod \sigma(0, 0, 2, -1)!! = 0 \text{ for all } j, \kappa, c \text{ values}$$

Evidently, this criterion generates false positives. For instance:

$$n_\omega(j, \kappa, c) \bmod \sigma(0, 0, 2, -1)!! = 0 \Rightarrow n_\omega(j, \kappa, c) \bmod \sigma(0, 0, 2, -3)!! = 0$$

Remark 33. False positives come in two flavors: “False false positives” and “True false positives”. $\sigma(0, 0, 2, -3)!!$ is a “false false positive”. Detecting $\sigma(0, 0, 2, -3)!!$ is useful as it decreases the target but allows progress. In the example above, instead of peeling-off $(2c-1)!!$ in one round, a first round will peel-off $(2c-3)!!$ and leave an extra $(2c-1)$ to some subsequent round. By opposition a “True false positive” is a candidate appearing as a factor of the target by the sole effect of chance over the available dataset.

To visualize efficiently execution we introduce *Backgammon diagrams*.

²⁴In our case LLL.

5.12.5 Backgammon diagrams

The gradient descent monitoring tool is called “Backgammon diagrams” because of its visual similarity to a backgammon board (Figure 3.11).

In a Backgammon diagram the x -axis shows a search performed over a coordinate u_s within an interval $u_s \in \{u_{\text{start}}, \dots, u_{\text{end}}\}$.

In all the following, consider that all \bar{u} coordinates other than s were fixed to correct values denoted by \checkmark .

The draughts represent experiments with fixed j, κ values and different c values. In the example $j = 11$, $\kappa = 6$ and $40 \leq c \leq 47$. Each draught color corresponds to a different c value ($\bullet, \color{blue}\bullet, \color{red}\bullet, \color{green}\bullet, \color{orange}\bullet, \color{purple}\bullet, \color{brown}\bullet$). The legend is not repeated to save space. Draughts were slightly lifted to avoid covering each other.

If for a given c value:

$$n_\omega(j, \kappa, c) \bmod \phi_i(\sigma(\checkmark, \dots, \checkmark, u_s, \checkmark, \dots, \checkmark)) = 0$$

then the draught of c 's color is lowered to the bottom of diagram, else it is raised to the top. Hence, a glance at the diagram shows which u_s values are compatible with the target $n_\omega(j, \kappa, c)$ for each c .

Two additional features enhance reading: a red line showing the correct answer²⁵ and little red triangles (\blacktriangle) denoting u_s values for which tests succeeded for all c values.

This view is grandly simplified with respect to reality. In the code j, κ vary as well (resulting in multi-dimensional and hence unvisualizable diagrams) and several u_s s are simultaneously tried at each round.

5.12.6 A worked-out example

We want to disassemble:

$$n_\omega(j, \kappa, c) = (2c - 1)!!^2 \cdot C_{\kappa-1} \cdot C_{\frac{j-3}{2}} \cdot (2\kappa - 1) \cdot (j - 2) \cdot \prod_{i=1}^{\frac{j-1}{2}} (2c - 2\kappa + 2i - 1)(\kappa - i + 1)$$

²⁵This is, of course, not provided by the software.

step	$\omega + 1$
target	$n_\omega(j, \kappa, c)$
candidate	$(uc - 1)!!$
	<p style="text-align: center;"> ● $c = 40$ ● $c = 41$ ● $c = 42$ ● $c = 43$ ● $c = 44$ ● $c = 45$ ● $c = 46$ ● $c = 47$ </p>
new target	$n_{\omega+1}(j, \kappa, c) = n_\omega(j, \kappa, c) / (2c - 1)!!$
database update	$\forall i \text{ do } t_i = t_i / (2c_i - 1)!!$
remarks	We have one solution which is $u = 2$.
step	illustrating option 1: $\omega + 2$
target	$n_{\omega+1}(j, \kappa, c)$
candidate	illustrating option 1: $(uc - 1)!!$
new target	$n_{\omega+2}(j, \kappa, c) = n_{\omega+1}(j, \kappa, c) / (2c - 1)!!$
database update	$\forall i \text{ do } t_i = t_i / (2c_i - 1)!!$
remarks	We have one solution which is $u = 2$.

step	illustrating option 2: $\omega + 2$
target	$n_{\omega+1}(j, \kappa, c)$
candidate	illustrating option 2: $(2c + u)!!$
new target	$n_{\omega+2}(j, \kappa, c) = n_{\omega+1}(j, \kappa, c) / (2c - 1)!!$
database update	$\forall i \text{ do } t_i = t_i / (2c_i - 1)!!$
remarks	4 solutions added to backtracking list.
step	illustrating option 1: $\omega + 3$
target	$n_{\omega+2}(j, \kappa, c)$
candidate	illustrating option 1: $(uc - 1)!!$
new target	None, wrong guess.
database update	None, wrong guess.
remarks	No solutions: repeat step $\omega + 3$ with another candidate.
step	illustrating option 2: $\omega + 3$
target	$n_{\omega+2}(j, \kappa, c)$
candidate	illustrating option 2: $(u\kappa - 1)$
new target	$n_{\omega+3}(j, \kappa, c) = n_{\omega+2}(j, \kappa, c) / (2\kappa - 1)$
database update	$\forall i \text{ do } t_i = t_i / (2\kappa_i - 1)$
remarks	7 solutions added to backtracking list.

step	illustrating option 3: $\omega + 3$
target	$n_{\omega+2}(j, \kappa, c)$
candidate	illustrating option 3: $(2\kappa + u)$
new target	$n_{\omega+3}(j, \kappa, c) = n_{\omega+2}(j, \kappa, c) / (2\kappa - 1)$
database update	$\forall i \text{ do } t_i = t_i / (2\kappa_i - 1)$
remarks	14 solutions added to backtracking list.
step	illustrating option 1: $\omega + 4$
target	$n_{\omega+3}(j, \kappa, c)$
candidate	illustrating option 1: $(j + u)$
new target	$n_{\omega+4}(j, \kappa, c) = n_{\omega+3}(j, \kappa, c) / (j - 2)$
database update	$\forall i \text{ do } t_i = t_i / (j_i - 2)$
remarks	13 solutions added to backtracking list.
step	illustrating option 2: $\omega + 4$
target	$n_{\omega+3}(j, \kappa, c)$
candidate	illustrating option 2: $(uj - 2)$
new target	$n_{\omega+4}(j, \kappa, c) = n_{\omega+3}(j, \kappa, c) / (j - 2)$
database update	$\forall i \text{ do } t_i = t_i / (j_i - 2)$
remarks	9 solutions added to backtracking list.

step	illustrating option 1: $\omega + 5$
target	$n_{\omega+4}(j, \kappa, c)$
candidate	illustrating option 1: $C_{\kappa+u}$
new target	$n_{\omega+5}(j, \kappa, c) = n_{\omega+4}(j, \kappa, c) / C_{\kappa-1}$
database update	$\forall i \text{ do } t_i = t_i / C_{\kappa_i-1}$
remarks	7 solutions added to backtracking list.
step	illustrating option 2: $\omega + 5$
target	$n_{\omega+4}(j, \kappa, c)$
candidate	illustrating option 2: $C_{u\kappa-1}$
new target	$n_{\omega+5}(j, \kappa, c) = n_{\omega+4}(j, \kappa, c) / C_{\kappa-1}$
database update	$\forall i \text{ do } t_i = t_i / C_{\kappa_i-1}$
remarks	2 solutions added to backtracking list.

step	illustrating option 1: $\omega + 6$
target	$n_{\omega+5}(j, \kappa, c)$
candidate	illustrating option 1: $\prod_{x=1}^{\frac{j-1}{2}} (2c + uk + 2x - 1)$
new target	$n_{\omega+6}(j, \kappa, c) = n_{\omega+5}(j, \kappa, c) / \prod_{x=1}^{\frac{j-1}{2}} (2c - 2k + 2x - 1)$
database update	$\forall i \text{ do } t_i = t_i / \prod_{x=1}^{\frac{j_i-1}{2}} (2c_i - 2k_i + 2x - 1)$
remarks	We have one solution which is $u = -2$.
step	illustrating option 2: $\omega + 6$
target	$n_{\omega+5}(j, \kappa, c)$
candidate	illustrating option 2: $\prod_{x=1}^{\frac{j-1}{2}} (2c - 2k + 2x + u)$
new target	$n_{\omega+6}(j, \kappa, c) = n_{\omega+5}(j, \kappa, c) / \prod_{x=1}^{\frac{j-1}{2}} (2c - 2k + 2x - 1)$
database update	$\forall i \text{ do } t_i = t_i / \prod_{x=1}^{\frac{j_i-1}{2}} (2c_i - 2k_i + 2x - 1)$
remarks	We have one solution which is $u = -1$.

step	illustrating option 3: $\omega + 6$
target	$n_{\omega+5}(j, \kappa, c)$
candidate	illustrating option 3: $\prod_{x=1}^{\frac{j-1}{2}} (uc - 2k + 2x - 1)$
new target	$n_{\omega+6}(j, \kappa, c) = n_{\omega+5}(j, \kappa, c) / \prod_{x=1}^{\frac{j-1}{2}} (uc - 2k + 2x - 1)$
database update	$\forall i \text{ do } t_i = t_i / \prod_{x=1}^{\frac{j_i-1}{2}} (uc_i - 2k_i + 2x - 1)$
remarks	We have one solution which is $u = 2$.
step	illustrating option 4: $\omega + 6$
target	$n_{\omega+5}(j, \kappa, c)$
candidate	illustrating option 4: $\prod_{x=1}^{\frac{2uj+j-1}{2}} (2c - 2\kappa + 2x - 1)$
new target	$n_{\omega+6}(j, \kappa, c) = n_{\omega+5}(j, \kappa, c) / \prod_{x=1}^{\frac{2uj+j-1}{2}} (2c - 2\kappa + 2x - 1)$
database update	$\forall i \text{ do } t_i = t_i / \prod_{x=1}^{\frac{2uj_i+j_i-1}{2}} (2c_i - 2\kappa_i + 2x - 1)$
remarks	We have one solution which is $u = 0$.

step	$\omega + 7$
target	$n_{\omega+6}(j, \kappa, c)$
candidate	$C_{\frac{j+2u+1}{2}}$
new target	$n_{\omega+7}(j, \kappa, c) = n_{\omega+6}(j, \kappa, c) / C_{\frac{j-3}{2}}$
database update	$\forall i \text{ do } t_i = t_i / C_{\frac{j_i-3}{2}}$
remarks	3 solutions added to backtracking list.
step	$\omega + 8$
target	$n_{\omega+7}(j, \kappa, c)$
candidate	$\prod_{x=1}^{\frac{j-1}{2}} (\kappa - x + u)$
new target	$n_{\omega+8}(j, \kappa, c) = n_{\omega+7}(j, \kappa, c) / \prod_{x=1}^{\frac{j-1}{2}} (\kappa - x + 1)$
database update	$\forall i \text{ do } t_i = t_i / \prod_{x=1}^{\frac{j_i-1}{2}} (\kappa_i - x + 1)$
remarks	4 solutions added to backtracking list.

step	$\omega + 8$
target	$n_{\omega+7}(j, \kappa, c)$
candidate	$\prod_{x=1}^{\frac{j-1}{2}} (\kappa + ux + 1)$
new target	$n_{\omega+8}(j, \kappa, c) = n_{\omega+7}(j, \kappa, c) / \prod_{x=1}^{\frac{j-1}{2}} (\kappa - x + 1)$
database update	$\forall i \text{ do } t_i = t_i / \prod_{x=1}^{\frac{j_i-1}{2}} (\kappa_i - x + 1)$
remarks	2 solutions added to backtracking list.

At this point one of the backtracking branches gives the constant function 1. $n_{\omega}(j, \kappa, c)$ was hence disassembled.

5.12.7 The Pathfinder

As we have just seen, at any step the algorithm can take several paths each of which offering a different backtracking fan-out. At first we considered resorting to a 2D-backtracking where one parameter is the offered fan-out and the second is the ϕ -backtracking *per se*. The bookkeeping associated to this procedure seemed prohibitive, therefore we just opted to take the candidate with the least fan-out at each step. This appeared sufficient for our purpose and compatible with the computational means at hand.

5.12.8 The Decimator

A very significant speed-up is achieved by a software module called “the Decimator”. The Decimator restricts the \vec{u} space by removing u_i affine combinations incompatible with the target.

Consider the target:

$$n_{\omega+2}(j, \kappa, c) = C_{\kappa-1} \cdot C_{\frac{j-3}{2}} \cdot (2\kappa - 1) \cdot (j - 2) \cdot \prod_{i=1}^{\frac{j-1}{2}} (2c - 2\kappa + 2i - 1)(\kappa - i + 1)$$

Create the database B given in Table 3.19.

Assume that we want to test a candidate $y(j, \kappa, c) = u_0j + u_1\kappa + u_2c + u_3$. We are typically interested in exploring each u_i over a small interval, e.g. $[-8, 8]$. Start by creating a list \mathcal{L} of all $(2 \times 8 + 1)^4$ \vec{u} -values.

For each $\{u_0, u_1, u_2, u_3\}$ value, if a t_i is not divisible by at least one $y(j_i, \kappa_i, c_i)$ then remove $\{u_0, u_1, u_2, u_3\}$ from \mathcal{L} .

Table 3.20 gives the number of \vec{u} s eliminated from $[-8, 8]^4$ by each c value.

i	j_i	κ_i	c_i	$t_i = n_{\omega+2}(j_i, \kappa_i, c_i)$
1	11	6	40	86562004597992000
2	11	6	41	99107222655672000
3	11	6	42	113065986409992000
4	11	6	43	128554477699032000
5	11	6	44	145695074725569600
6	11	6	45	164616513001617600
7	11	6	46	185454046292961600
8	11	6	47	208349607563697600

Table 3.19: The database B

c	40	41	42	43	44	45	46	47
eliminated \vec{u}s	51721	54371	55635	56161	52771	51203	51609	46383

Table 3.20: Number of eliminated \vec{u} combinations per c tried.

When we merge all the forbidden \vec{u} s, removing duplicates, we get a collection of 78567 combinations to skip. We are hence left with $(2 \times 8 + 1)^4 - 78567 = 4954$ survivors to test, i.e. 5.9%.

The above example is restricted to eight c toy-values. In reality we perform calculations on $\simeq 10^5$ combinations of (j, κ, c) . This reduces drastically the search space²⁶. Note however that as c increases the percentage of newly removed \vec{u} values per round decreases.

Note that, in practice even an exploration in $[-3, 3]^4$ would have sufficed²⁷. For $\vec{u} \in [-3, 3]^4$ we get 332 survivors out of 2401 (14%).

For $[-8, 8]^4$ and $40 \leq c \leq 47$, decimating for $y(j, \kappa, c) = C_{u_0j+u_1\kappa+u_2c+u_3}$ removes 83233 candidates, leaving only 288 possibilities (0.35%).

Decimating over $[-5, 5]^5$ and $20 \leq c \leq 60$ for the candidate:

$$y(j, \kappa, c) = \prod_{i=1}^{\frac{j-1}{2}} (u_0j + u_1\kappa + u_2c + u_3 + u_4i)$$

results in 1496 survivors out of 161051 (0.93%).

Counter-intuitively, the more “complex”²⁸ the candidate is the more efficiently it is decimated. We hence select the candidates in decreasing complexity order. Measuring the complexity of mathematical formulas can be a subjective task, as it depends on various factors such as the number of terms, the presence of functions, exponents, and variables, as well as the overall structure. There’s no definitive metric to quantify formula complexity. The following criteria can nonetheless be used to “measure” complexity and hence get rid of ϕ_i s as fast as possible:

Counting Elements: We can count the number of distinct elements in each candidate, such as variables, constants, operators, and functions. For example, the candidate $\kappa - 1$ has two elements while in others we may have multiple variables, exponents, a product symbol, and a summation, which increases its complexity.

²⁶ e.g., exploring for $20 \leq c \leq 60$ reduces the survivors’ pool to 3702 (4.4%).

²⁷ The 7 in ϕ_8 would have been decimated in two rounds.

²⁸ i.e., closer to a random oracle.

Nesting and Hierarchy: Analyze the nesting of operations and functions within the candidates. A candidate with multiple levels of nesting or hierarchy can be regarded as more complex.

Mathematical Operations: Consider the types of mathematical operations present in the candidates. More complex operations, such as exponentiation and summation, contribute to higher complexity compared to simpler operations like addition or multiplication.

Function Complexity: If the candidate includes functions, their complexity should be taken into account. For instance, a Catalan number can add complexity compared to linear or constant functions.

Symbolic Representation: Represent each candidate in a symbolic format, such as a parse tree or abstract syntax tree. Compare the depth and branching of the trees as a rough measure of complexity.

Information Theory: Explore concepts from information theory, such as Kolmogorov complexity or algorithmic information theory, to quantify the amount of information needed to describe each candidate. This approach can be quite theoretical and may not provide a practical measure for all cases.

We resorted to a much more brutal approach [Bana, 2021]. Using the API²⁹, we asked GPT-4 to compare candidates pairwise, obtained a subjective complexity comparison (\prec) of each pair and translated the ternary results³⁰ to a directed graph. Because the LLM does not provide consistent answers (i.e. it might say that $A \prec B \prec C$ and... $C \prec A$) we performed a random walk of 10^6 steps on the graph and counted the number of times each candidate was visited. The most visited candidate was considered as the “most complex”. We then removed this candidate from the graph and started over again.

As a final note, we remark that some ϕ_i s are “process killers”. This is the case of candidates such as $(2c)!$. We called ϕ_i s silencers “opioids” are they efficiently remove the symptoms but do not remove any lurgy. Because, in essence, $(2c)!$ contains just about any number of interest, it silences the modular tests. We hence start by launching the process with $\bar{u}!$ and start the backtracking afresh for each $n_0(j, \kappa, c) / \bar{u}!$. Luckily, the early investigation of [Bosnia & Herzegovina](#) by which we started allowed to uncover the opioids before formulae get too complex.

5.12.9 Ascending to descend

As underlined in [Sanderson, 1979], some descents require intermediate ascents. The process described so far advances only in the case of a monotonous descent. In other words, if the initial ϕ_i s were wrongly chosen the process will not converge.

We thus need a process allowing temporary ascents to get out of bowls. To that end we use a modified version of Broyden-Fletcher-Goldfarb-Shanno’s (BFGS) algorithm. This requires a more refined measure of the penalty/profit of each move which cannot just be the Boolean “ x divides y ”.

Denote by p_i the i -th prime.

We introduce a measure called “brittleness”³¹ denoted $\Xi(n)$.

$$\text{Let: } n = \prod_{i=0}^{a-1} p_i^{m_i} \in \mathbb{Q} \text{ be a simplified fraction. Then } \Xi(n) = \sum_{i=0}^{a-1} |m_i|$$

²⁹ endpoint <https://api.openai.com/v1/engines/davinci-codex/completions>

³⁰“ A is more complex than B ”: $A \rightarrow B$, “ B is more complex than A ”: $B \rightarrow A$ or “unsure”: no edge.

³¹Brittleness is a generalization the prime Ω function to \mathbb{Q} : $\Xi(n) = \Omega(\text{numerator}(n)) + \Omega(\text{denominator}(n))$.

In other words, $\Xi(n)$ counts, with repetition, the number of distinct factors appearing in either the numerator or the denominator of n .

The following example illustrates the evolution of brittleness (y -axis) during the peeling-off process.

Dotted lines \Rightarrow Target's brittleness at the concerned j, κ, c points.

Draughts \Rightarrow Raised to $\Xi(\text{target/candidate})$ for different u values.

As before, \blacktriangle denotes us at which all draughts are lower than their same-color dotted lines. The thin red vertical line is the correct answer.

As an example start with the target:

$$n_1(j, \kappa, c) = C_{\kappa-1} \cdot C_{\frac{j-3}{2}} \cdot (2\kappa-1) \cdot (j-2) \cdot \prod_{i=1}^{\frac{j-1}{2}} (2c-2\kappa+2i-1)(\kappa-i+1)$$

Figure 3.13 represents the following steps:

step	1 of Figure 3.13
target	$n_0(j, \kappa, c)$
candidate	illustrating first option: $u\kappa-1$
new target	$n_1(j, \kappa, c) = n_0(j, \kappa, c)/(2\kappa-1)$

step	1' of Figure 3.13
target	$n_0(j, \kappa, c)$
candidate	illustrating second option: $2\kappa+u$
new target	$n_1(j, \kappa, c) = n_0(j, \kappa, c)/(2\kappa-1)$

step	2 of Figure 3.13
target	$n_1(j, \kappa, c)$
candidate	$uj-2$
new target	$n_2(j, \kappa, c) = n_1(j, \kappa, c)/(j-2)$

step	3 of Figure 3.13
target	$n_2(j, \kappa, c)$
candidate	illustrating first option: $\prod_{x=1}^{\frac{j-1}{2}} (2c-2k+ux-1)$
new target	$n_3(j, \kappa, c) = n_2(j, \kappa, c) / \prod_{x=1}^{\frac{j-1}{2}} (2c-2k+2x-1)$

step	3' of Figure 3.13
target	$n_2(j, \kappa, c)$
candidate	illustrating second option: $\prod_{x=1}^{\frac{j+2u+1}{2}} (2c - 2k + 2x - 1)$
new target	$n_3(j, \kappa, c) = n_2(j, \kappa, c) / \prod_{x=1}^{\frac{j-1}{2}} (2c - 2k + 2x - 1)$

step	4 of Figure 3.13
target	$n_3(j, \kappa, c)$
candidate	$C_{u\kappa-1}$
new target	$n_4(j, \kappa, c) = n_3(j, \kappa, c) / C_{\kappa-1}$

step	5 of Figure 3.13
target	$n_4(j, \kappa, c)$
candidate	$C_{\frac{j+2u+1}{2}}$
new target	$n_5(j, \kappa, c) = n_4(j, \kappa, c) / C_{\frac{j-3}{2}}$

step	6 of Figure 3.13
target	$n_5(j, \kappa, c)$
candidate	$\prod_{x=1}^{\frac{j-1}{2}} (k - x + u)$
new target	None: Processed finished.

5.13 Tables of Constants

a_0	a_1	a_2	$P(n)/(-2n)$	$T(n) - 3n^2$
-7351344	-32375839	46558512	$n(3+n)(17+n)$	$72 + 43n$
-2450448	-1768477	2450448	$n(1+n)(17+n)$	$36 + 39n$
-1081080	-16147379	23279256	$n(5+n)(15+n)$	$96 + 43n$
-793800	-232217	322560	$(4+n)(9+n)^2$	$100 + 39n$
-504504	-16140515	23279256	$n(7+n)(13+n)$	$112 + 43n$
-436590	7989199	-11531520	$(3+n)(9+n)(11+n)$	$120 + 43n$
-180180	-1001393	1441440	$(2+n)(5+n)(13+n)$	$84 + 39n$
-174636	-8069449	11639628	$n(9+n)(11+n)$	$120 + 43n$
-72072	-850133	1225224	$n(5+n)(13+n)$	$84 + 39n$
-72072	-251099	360360	$n(3+n)(13+n)$	$56 + 35n$
-72072	-52279	72072	$n(1+n)(15+n)$	$32 + 35n$
-45045	124048	-180180	$(1+n)(3+n)(13+n)$	$56 + 35n$
-28028	-999391	1441440	$(2+n)(7+n)(13+n)$	$112 + 43n$
-27720	-20417	27720	$n(1+n)(11+n)$	$24 + 27n$
-22050	27649	-40320	$(3+n)(7+n)(9+n)$	$80 + 35n$
-19305	424423	-612612	$(1+n)(5+n)(15+n)$	$96 + 43n$
-17640	-250007	360360	$n(7+n)(9+n)$	$80 + 35n$
-14700	-153907	221760	$(2+n)(7+n)(9+n)$	$80 + 35n$
-10395	124741	-180180	$(1+n)(5+n)(11+n)$	$72 + 35n$
-9450	76691	-110880	$(1+n)(5+n)(9+n)$	$60 + 31n$
-7350	-62563	90090	$n(7+n)^2$	$64 + 31n$
-7007	424566	-612612	$(1+n)(7+n)(13+n)$	$112 + 43n$
-6300	-14087	20160	$(2+n)(5+n)(9+n)$	$60 + 31n$
-6006	99839	-144144	$(1+n)(5+n)(13+n)$	$84 + 39n$
-4900	-21043	30240	$(2+n)(7+n)^2$	$64 + 31n$
-2520	-1879	2520	$n(1+n)(9+n)$	$20 + 23n$
-2450	28781	-41580	$(1+n)(7+n)^2$	$64 + 31n$
-1225	367	-560	$(3+n)(7+n)^2$	$64 + 31n$
-525	-2413	3465	$n(5+n)(7+n)$	$48 + 27n$
-450	1151	-1680	$(1+n)(5+n)^2$	$36 + 23n$
-350	1739	-2520	$(1+n)(5+n)(7+n)$	$48 + 27n$
-180	-299	420	$n(3+n)(5+n)$	$24 + 19n$
-105	142	-210	$(1+n)(3+n)(7+n)$	$32 + 23n$
-18	7	-12	$(1+n)(3+n)^2$	$16 + 15n$
-6	-5	6	$n(1+n)(3+n)$	$8 + 11n$

Table 3.21: Examples of convergence to $\frac{a_0}{a_1 + a_2 \log 2}$

a_0	a_1	a_2	$P(n)/(-2n)$	$T(n) - 3n^2$
1	1	-1	$\frac{n(1+n)^2}{n(3+n)^2}$	$4+7n$
9	11	-15	$\frac{n(3+n)^2}{n(5+n)^2}$	$16+15n$
50	147	-210	$\frac{n(5+n)^2}{n(1+n)(5+n)}$	$36+23n$
60	47	-60	$\frac{n(1+n)(5+n)}{(1+n)(3+n)(5+n)}$	$12+15n$
90	-79	120	$\frac{n(1+n)(7+n)}{n(3+n)(7+n)}$	$24+19n$
420	319	-420	$\frac{n(3+n)(7+n)}{(2+n)(5+n)^2}$	$16+19n$
420	887	-1260	$\frac{n(3+n)(7+n)}{(2+n)(5+n)^2}$	$32+23n$
900	361	-480	$\frac{(2+n)(5+n)^2}{(1+n)(3+n)(9+n)}$	$36+23n$
1890	-3443	5040	$\frac{(1+n)(3+n)(9+n)}{(2+n)(5+n)(7+n)}$	$40+27n$
2100	2377	-3360	$\frac{(2+n)(5+n)(7+n)}{n(5+n)(11+n)}$	$48+27n$
5544	50035	-72072	$\frac{n(5+n)(11+n)}{(1+n)(7+n)(11+n)}$	$72+35n$
6468	-249713	360360	$\frac{(1+n)(7+n)(11+n)}{n(3+n)(9+n)}$	$96+39n$
7560	19409	-27720	$\frac{n(3+n)(9+n)}{(1+n)(3+n)(11+n)}$	$40+27n$
8316	-19031	27720	$\frac{(1+n)(3+n)(11+n)}{(1+n)(3+n)(15+n)}$	$48+31n$
15444	-49705	72072	$\frac{(1+n)(3+n)(15+n)}{(1+n)(7+n)(9+n)}$	$64+39n$
22050	-499279	720720	$\frac{(1+n)(7+n)(9+n)}{(3+n)(7+n)(11+n)}$	$80+35n$
24255	-76586	110880	$\frac{(3+n)(7+n)(11+n)}{(2+n)(5+n)(15+n)}$	$96+39n$
25740	200107	-288288	$\frac{(2+n)(5+n)(15+n)}{n(5+n)(9+n)}$	$96+43n$
37800	250427	-360360	$\frac{n(5+n)(9+n)}{n(7+n)(11+n)}$	$60+31n$
38808	849671	-1225224	$\frac{n(7+n)(11+n)}{(1+n)(9+n)^2}$	$96+39n$
39690	-1997851	2882880	$\frac{(1+n)(9+n)^2}{(2+n)(5+n)(11+n)}$	$100+39n$
41580	154327	-221760	$\frac{(2+n)(5+n)(11+n)}{(2+n)(9+n)(11+n)}$	$72+35n$
58212	3997025	-5765760	$\frac{(2+n)(9+n)(11+n)}{n(9+n)^2}$	$120+43n$
79380	2123957	-3063060	$\frac{n(9+n)^2}{n(3+n)(11+n)}$	$100+39n$
83160	251561	-360360	$\frac{n(3+n)(11+n)}{(1+n)(9+n)(11+n)}$	$48+31n$
87318	-8491859	12252240	$\frac{(1+n)(9+n)(11+n)}{(2+n)(7+n)(11+n)}$	$120+43n$
97020	1999321	-2882880	$\frac{(2+n)(7+n)(11+n)}{(2+n)(9+n)^2}$	$96+39n$
132300	3997907	-5765760	$\frac{(2+n)(9+n)^2}{(3+n)(9+n)^2}$	$100+39n$
198450	-1227581	1774080	$\frac{(3+n)(9+n)^2}{n(3+n)(15+n)}$	$100+39n$
216216	852707	-1225224	$\frac{n(3+n)(15+n)}{n(1+n)(13+n)}$	$64+39n$
360360	263111	-360360	$\frac{n(1+n)(13+n)}{(3+n)(7+n)(13+n)}$	$28+31n$
630630	-3990557	5765760	$\frac{(3+n)(7+n)(13+n)}{(1+n)(3+n)(17+n)}$	$112+43n$
918918	-3383801	4900896	$\frac{(1+n)(3+n)(17+n)}{(4+n)(9+n)(11+n)}$	$72+43n$
1746360	2475007	-3548160	$\frac{(4+n)(9+n)(11+n)}{n(1+n)(19+n)}$	$120+43n$
46558512	33464927	-46558512	$\frac{n(1+n)(19+n)}{n(1+n)(19+n)}$	$40+43n$

Table 3.22: Examples of convergence to $\frac{a_0}{a_1+a_2 \log 2}$. The entry in blue is the one reported by the Ramanujan Project.

a_0	a_1	a_2	$P(n)/(-2n)$	$T(n) - 3n^2$
-80281600	-10675439	9459450	$(2+n)(3+n)(14+n)$	$45 + 35n$
-15728640	392683	-727650	$(4+n)^2(12+n)$	$65 + 35n$
-13107200	-263867	-28350	$(4+n)(5+n)(10+n)$	$55 + 31n$
-10485760	-93699	-103950	$(4+n)(5+n)(12+n)$	$65 + 35n$
-7372800	-884203	727650	$(2+n)(3+n)(12+n)$	$39 + 31n$
-6291456	149419	-257250	$(4+n)(8+n)^2$	$81 + 35n$
-5242880	-86807	9450	$(5+n)(6+n)(10+n)$	$77 + 35n$
-3932160	-116317	3150	$(4+n)(5+n)(8+n)$	$45 + 27n$
-3276800	-158859	22050	$(2+n)(5+n)(10+n)$	$33 + 27n$
-2621440	-48609	-1050	$(5+n)(6+n)(8+n)$	$63 + 31n$
-2359296	-168445	103950	$(2+n)(4+n)(12+n)$	$39 + 31n$
-491520	50593	-66150	$(3+n)(4+n)(10+n)$	$55 + 31n$
-327680	-21271	9450	$(2+n)(4+n)(10+n)$	$33 + 27n$
-230400	-1909	-22050	$(2+n)^2(5+n)$	$9 + 11n$
-196608	-184547	198450	$(3+n)(6+n)(10+n)$	$77 + 35n$
-163840	-4981	-22050	$n(5+n)(8+n)$	$9 + 19n$
-131072	-2951	-630	$(4+n)^2(8+n)$	$45 + 27n$
-122880	-13079	9450	$(2+n)(3+n)(10+n)$	$33 + 27n$
-61440	-2467	-3150	$(2+n)(4+n)(5+n)$	$15 + 15n$
-61440	791	-9450	$n(5+n)(6+n)$	$7 + 15n$
-51200	2839	-9450	$n(4+n)(5+n)$	$5 + 11n$
-49152	-1919	90	$(4+n)^2(6+n)$	$35 + 23n$
-36864	-2693	-450	$(2+n)(4+n)(6+n)$	$21 + 19n$
-18432	-419	-3150	$n(4+n)(6+n)$	$7 + 15n$
-18432	-419	-3150	$n(3+n)(8+n)$	$9 + 19n$
-8192	-487	-54	$(4+n)^3$	$25 + 19n$
-3072	121	-630	$n(4+n)^2$	$5 + 11n$
-2048	-129	-90	$(2+n)(4+n)^2$	$15 + 15n$
-2048	-43	-30	$(3+n)(4+n)(6+n)$	$35 + 23n$
-768	-77	-18	$(2+n)(3+n)(4+n)$	$15 + 15n$
-288	31	-90	$n(2+n)(3+n)$	$3 + 7n$

Table 3.23: Examples of convergence to $\frac{a_0}{a_1+a_2G}$

a_0	a_1	a_2	$P(n)/(-2n)$	$T(n) - 3n^2$
192	13	18	$(2+n)^2(3+n)$	$9+11n$
384	1	90	$n(3+n)(4+n)$	$5+11n$
2304	389	450	$n(3+n)(6+n)$	$7+15n$
3072	179	-18	$(3+n)(4+n)^2$	$25+19n$
4608	133	450	$(2+n)^2(4+n)$	$9+11n$
4608	383	-90	$(2+n)(3+n)(6+n)$	$21+19n$
11520	-1373	3150	$n(2+n)(4+n)$	$3+7n$
12288	973	-750	$(3+n)(6+n)^2$	$49+27n$
12288	1145	-630	$(2+n)(3+n)(8+n)$	$27+23n$
16384	-543	1050	$(3+n)(4+n)(8+n)$	$45+27n$
81920	3983	1350	$(4+n)^2(5+n)$	$25+19n$
89600	-10891	22050	$n(2+n)(5+n)$	$3+7n$
98304	2263	150	$(4+n)(6+n)^2$	$49+27n$
98304	35389	-36750	$(3+n)(6+n)(8+n)$	$63+31n$
122880	6563	3150	$(2+n)(5+n)(6+n)$	$21+19n$
147456	21365	22050	$n(4+n)(8+n)$	$9+19n$
163840	6789	450	$(4+n)(5+n)(6+n)$	$35+23n$
262144	710401	-771750	$(3+n)(8+n)^2$	$81+35n$
294912	18013	-3150	$(2+n)(4+n)(8+n)$	$27+23n$
524288	27787	-22050	$(4+n)(6+n)(10+n)$	$77+35n$
786432	19099	-5250	$(4+n)(6+n)(8+n)$	$63+31n$
983040	25979	-450	$(5+n)(6+n)^2$	$49+27n$
1310720	1723	28350	$(4+n)^2(10+n)$	$55+31n$
2949120	168821	22050	$(2+n)(5+n)(8+n)$	$27+23n$
9830400	-1833409	2182950	$(3+n)(4+n)(12+n)$	$65+35n$
12582912	184025	-7350	$(5+n)(8+n)^2$	$81+35n$
39321600	1965547	-727650	$(2+n)(5+n)(12+n)$	$39+31n$
165150720	12969199	-9459450	$(2+n)(4+n)(14+n)$	$45+35n$
330301440	17687791	-9459450	$(2+n)(5+n)(14+n)$	$45+35n$

Table 3.24: Examples of convergence to $\frac{a_0}{a_1+a_2G}$

c	a_0	a_1	a_2
3	2	1	0
4	1	1	-1
5	-1	-3	4
6	-2	-17	24
7	-3	-67	96
8	12	667	-960
9	5	666	-960
10	30	9319	-13440
11	-105	-74537	107520
12	-280	-447187	645120
13	126	447173	-645120
14	63	491884	-709632
15	231	3935051	-5677056
16	2772	102311095	-147603456
17	-1287	-102310996	147603456
18	-6006	-1023109531	1476034560
19	45045	16369749493	-23616552960
20	720720	556571437717	-802962800640

Table 3.25: Examples of convergence to $\frac{a_0}{a_1+a_2 \log 2}$ for R_c

j	α_j	β_j
3	-1	4
5	19	234
7	5818/3	254456/3
9	667115	60003486
11	467946090	71121907440
13	554143204110	127451285438100
15	994115449382940	322092692148962160
17	2516347061651130075	1092094185270706446150
19	8546069024090201027250	4785798287838257081935200
21	37508692924557081882027450	26331102038134635548392485900
23	206659254109760483703789089700	177726957997323983116663150902000
25	1396637676485497608584841260027550	1444123356588023432320434243315206700
27	11361110319787394788017568214856502500	13905999029609441333101619589964946580000
29	109509742351999832489255793094925601037500	156598931559029451368898717824937174831465000
31	1234320809247763942235545044494798498436195000	2039097976865181167119056627863149102390546140000
33	16085205915675471439195309128783843538512283666875	30401039180587356456007967587920548312623820610393750
35	239989379884263177615577263747245812249369757283461250	514537230471714428505965482811829861362838523445500920000

j	$\bar{\alpha}_j$	$\bar{\beta}_j$
3	-1/3	-14/3
5	-17	-8
7	-758	-27820
9	-302117	-23010044
11	-1091480994/5	-146282046156/5
13	-262476468810	-54596049230880
15	-475443072646380	-141682352738003640
17	-1211573031414907725	-489475664504671450500
19	-4135193781750207709650	-2175112041708995560914300
21	-18218507239728799899288030	-12097088912487772715204794320
23	-100680148628028059378172563700	-82356361704096372069838207986600
25	-682078864161239229949889893754850	-673893917980353010760236819146271800
27	-5559692282317104149119150499246482500	-6527078739785105011529098668023829975000
29	-53681246247288656939970174534335708392500	-73865394837022289182570623863734010339760000
31	-605939306349175124039948450466713432304279000	-965867254322525126192328035746702493817188406000
33	-7906287653442943862409767973858652552097126548125	-14452693830499521315903006473321900713406050369972500
35	-118090012323922699712409299094969252935070156336941250	-245391045609131483699190960852072336578359955374403917500

Table 3.26: The first $\bar{\alpha}_j = \left\{ \left\{ \alpha_j^{\alpha}, \alpha_j^{\beta} \right\} \right\}$ values and the first $\bar{\beta}_j = \left\{ \left\{ \beta_j^{\alpha}, \beta_j^{\beta} \right\} \right\}$ values.

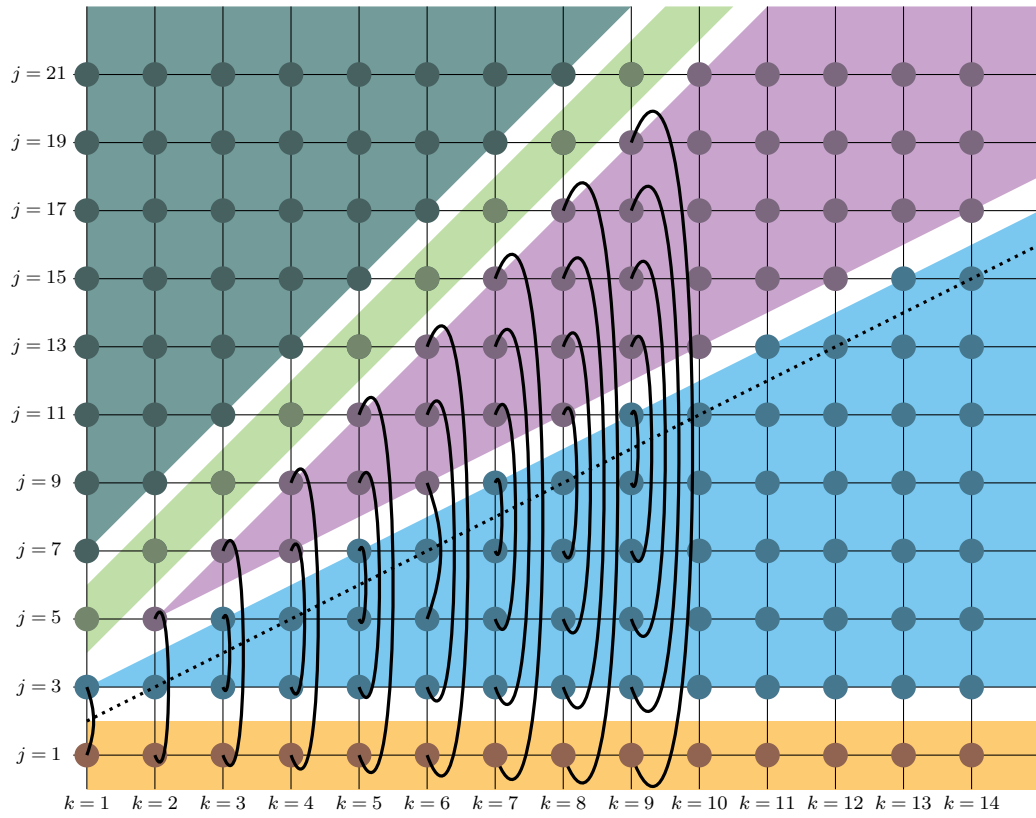


Figure 3.1: The Five j, k Areas. The meaning of the arches connecting **Kosovo**, **Serbia**, and **Montenegro** will be clarified later.

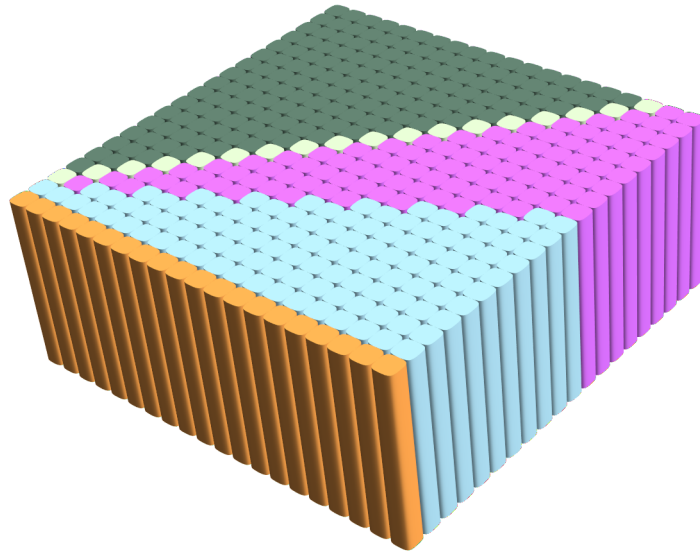


Figure 3.2: The areas of Figure 3.1, with the dimension c added. Each point in space corresponds to a $Q_{j,\kappa,c}$ value.

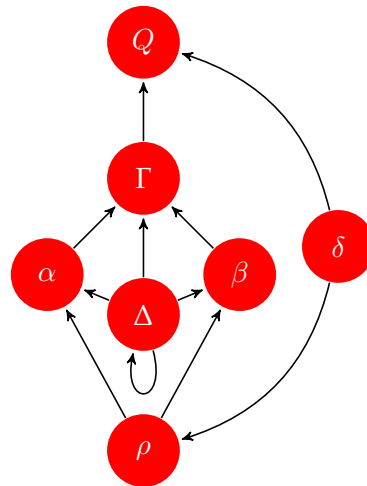


Figure 3.3: Functional dependency between the functions computing $Q_{1,\kappa,c}$

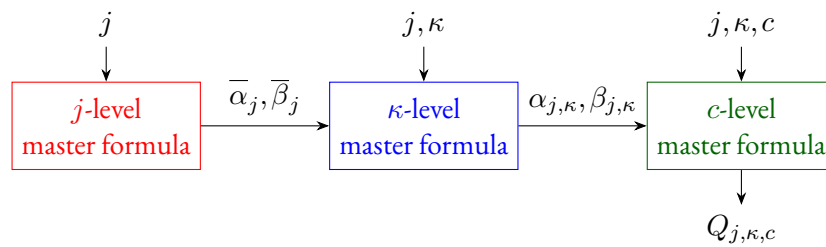


Figure 3.4: j , κ and c -level master formulae. This paper provides exact algebraic processes for inferring $Q_{j,\kappa,c}$ for all areas.

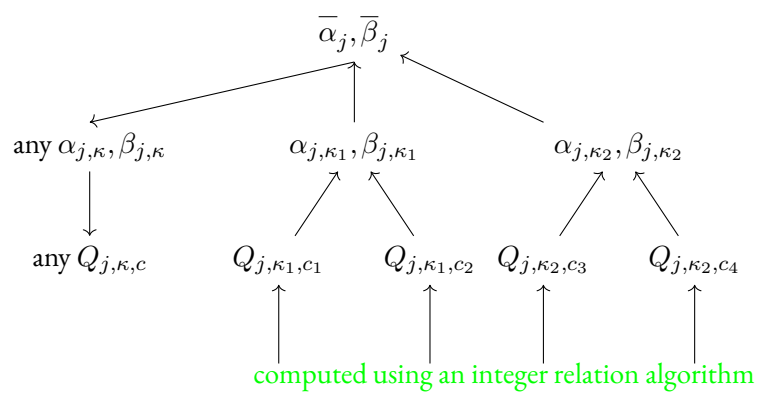


Figure 3.5: κ -level resolution process for Kosovo.

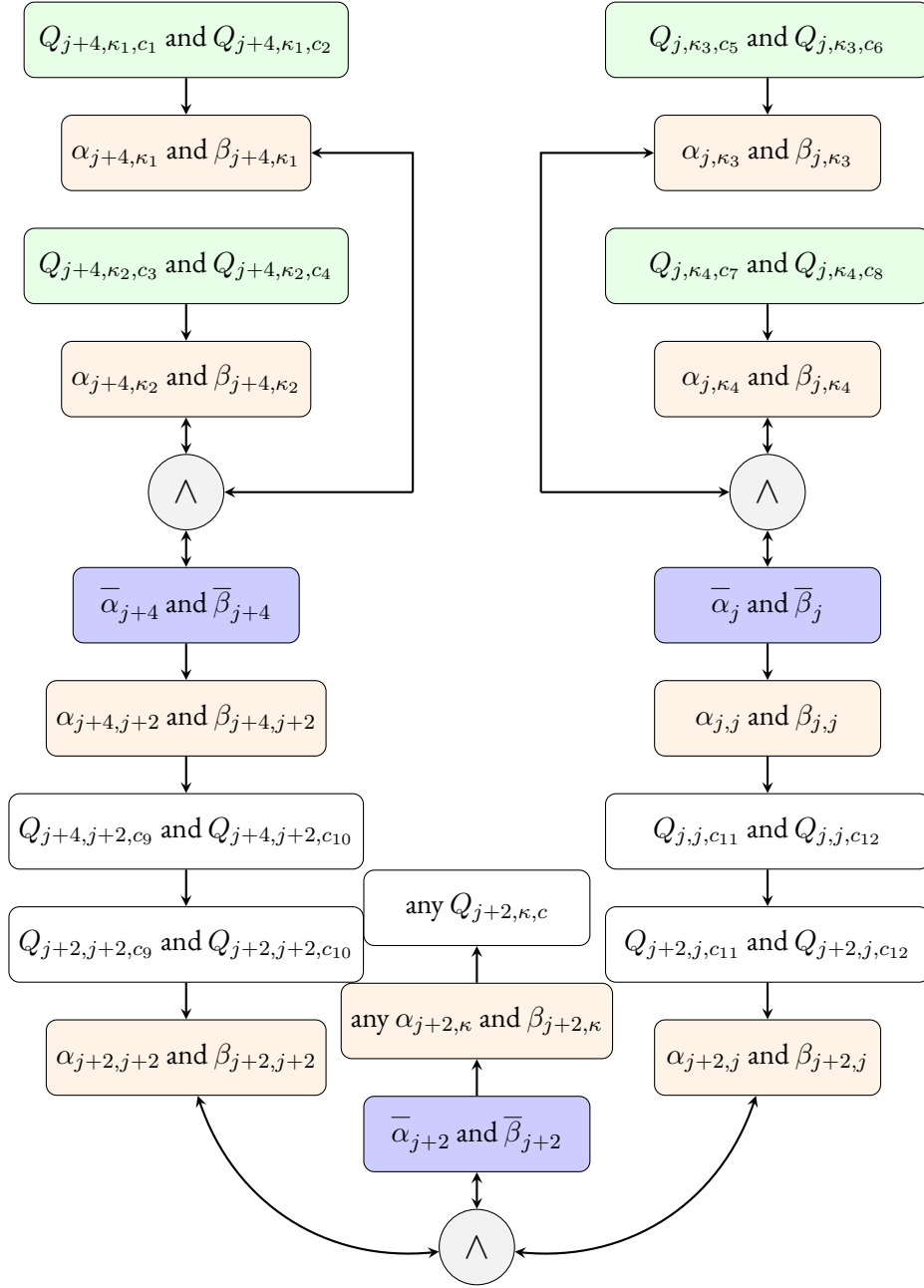


Figure 3.6: Inferring $\bar{\alpha}_{j+2}, \bar{\beta}_{j+2}$ from $\bar{\alpha}_j, \bar{\beta}_j$ and $\bar{\alpha}_{j+4}, \bar{\beta}_{j+4}$. The $Q_{j, \kappa, c}$ in green boxes are determined using integer relation algorithms.



Figure 3.7: $Q_{j,\kappa,c}$ *Requiescat in pace.*

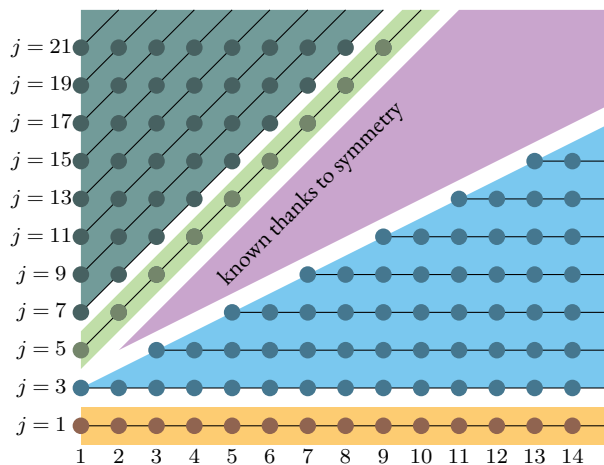


Figure 3.8: The axes along which the κ -level master formulae operate in each area. We hence see that a finite amount of “magic” information in one dimension (that we fully provide in this paper) allows to algebraically compute $Q_{j,\kappa,c}$ over the two remaining dimensions.

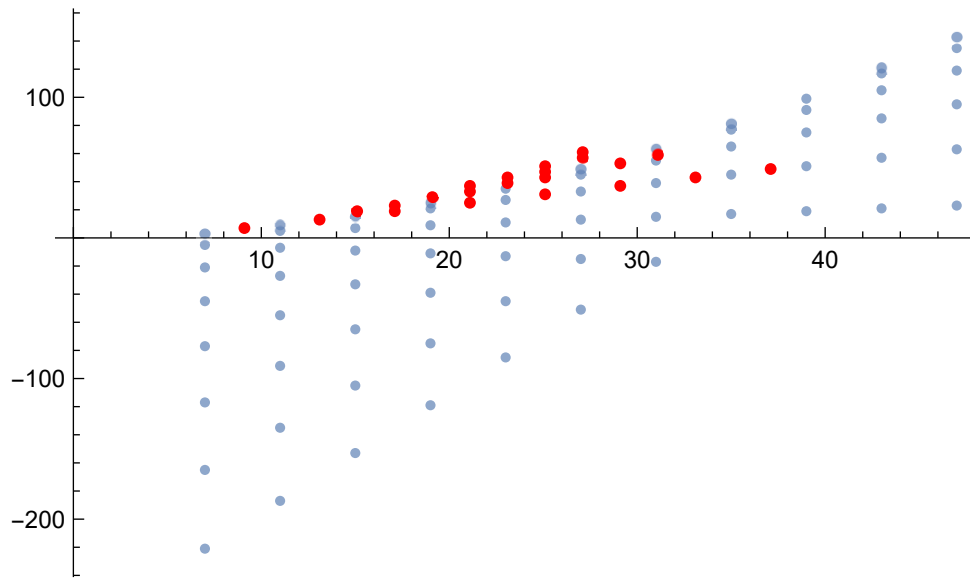


Figure 3.9: ϵ, δ for the Balkans (in blue) and Inostranstvo (in red).

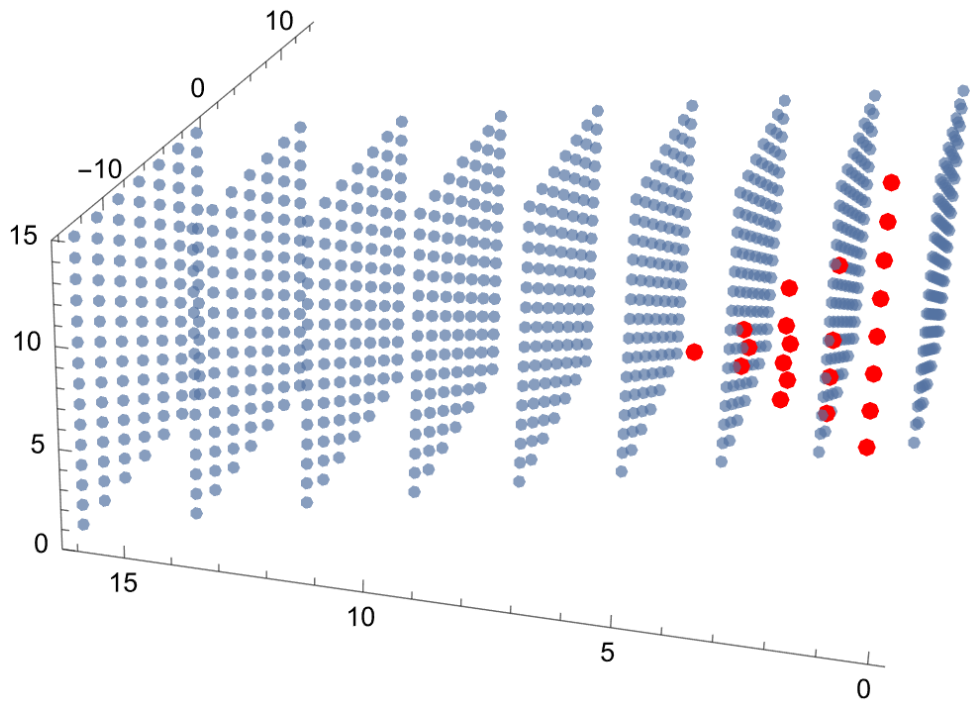


Figure 3.10: τ, η, μ for the Balkans (in blue) and Inostranstvo (in red).

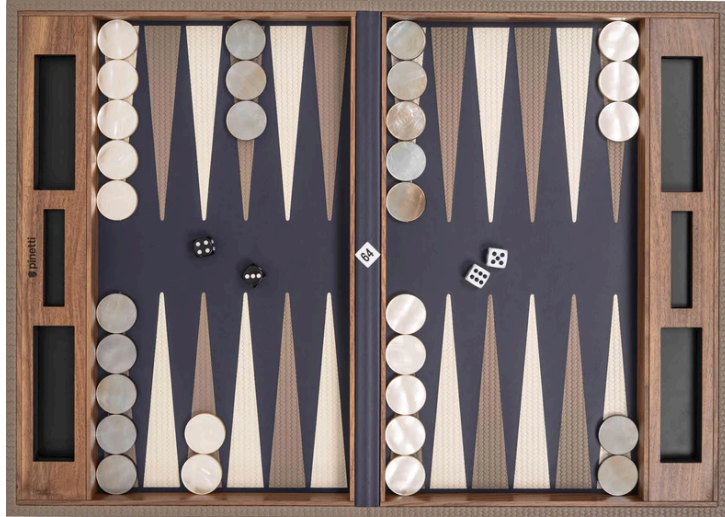


Figure 3.11: A typical Backgammon board.



Figure 3.12: $Q_{j,\kappa,c}$ Process killers.

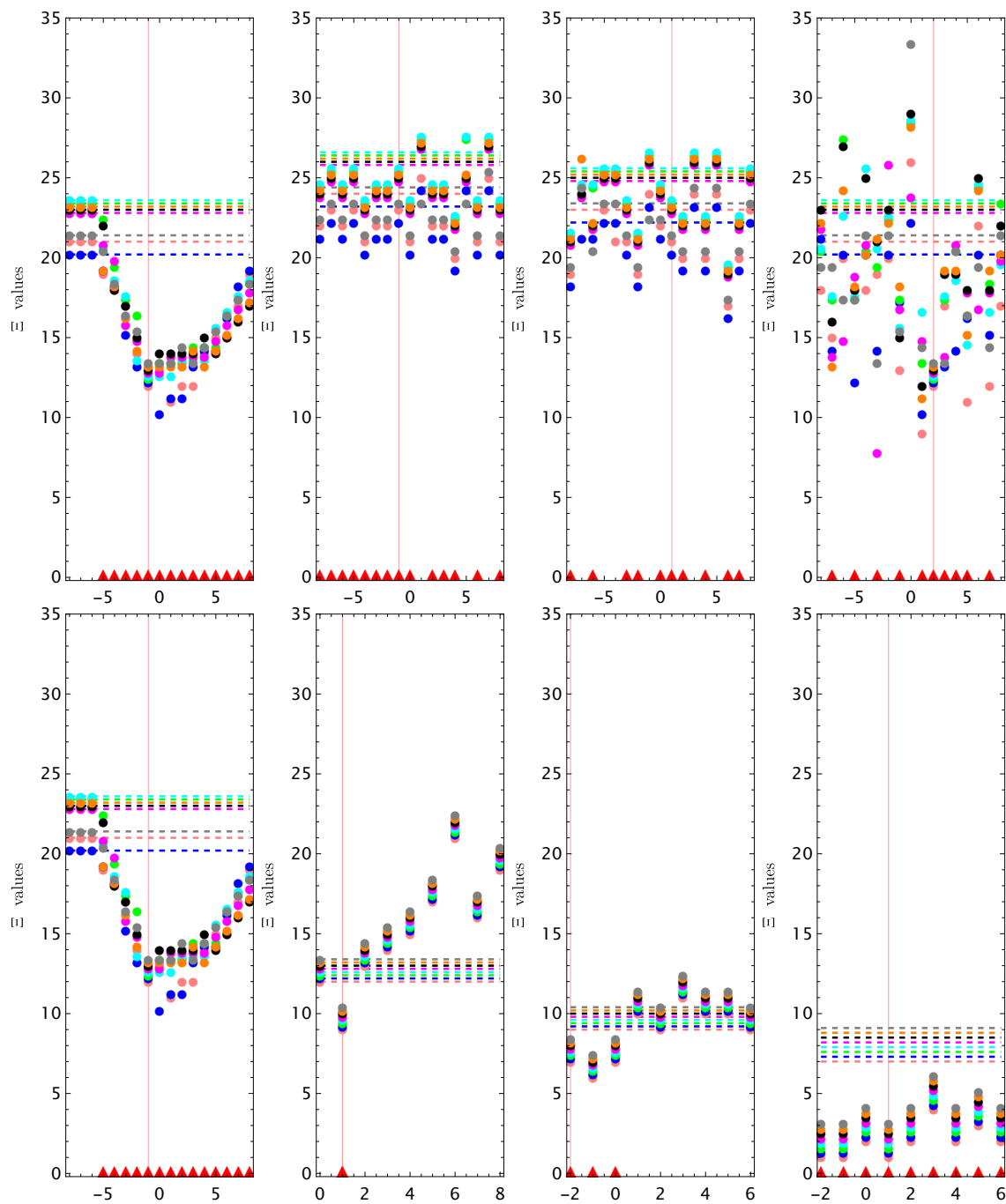


Figure 3.13: Steps are ordered from left→right and up→down, i.e. $\begin{matrix} 1 & 1' & 2 & 3 \\ 3' & 4 & 5 & 6 \end{matrix}$. Note the steady decrease of Ξ levels. Steps 1 and 1' are two division options occurring at the same step and so are 3 and 3'.

5.14 Mathematica Code

```

1 (* code snippet "1. Montenegro" *)
2 NumQ[k_,c_]:=2k+1+ContinuedFractionK[-2n^(n+2k)(n+c).3n^2+(3+4k)n+2k+1,{n,1,5000}];
3 vD[a_,b_,k_,c_]:=If[c<2,a+b*c,-2c(2c-1)(2c-k-1)^2vD[a,b,k,c-2]+(8c^2+(2-8k)c-2k+1)vD[a,b,k,c-1]];
4 vG[a_,b_,k_,c_]:=((2c-1)!)^2Catalan+Product[(2(c-j)-1),{j,0,k-1}]*vD[a,b,k,c-1];
5 vd[k_]:=4^(k-1)/(2k-1)/CatalanNumber[k-1];vr[k_]:=vd[k]*(-1)^(k-1-2k)/((2k-3)!!);
6 va[k_]:=vr[k]vD[1,-2,1,k-1];vb[k_]:=-vr[k](2k-3)^2vD[1,12,2,k-1]-va[k];
7 QFor[k_,c_]:=vd[k](2c)!/vG[va[k],vb[k],k,c];
8 Print[Union[Flatten[Table[N[QFor[k,c],200],{c,1,14}],{k,1,14}]]][[1]];ClearAll["Global`*"];

```

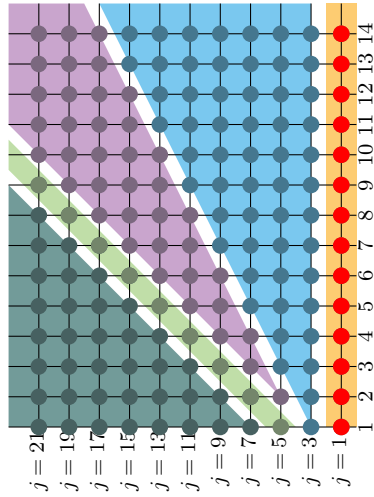


Figure 3.14: The points on which **Montenegro** is tested for $1 \leq c \leq 14$ by the snippet "1. Montenegro" are shown in red.

```

1 (* code snippet "2. Bosnia" *)
2 NumQ[j_,c_]:= -j+ContinuedFractionK[-2n(n-2)(n+c)(n+j-1).3n^2+(2j-3)n-j,{n,1,5000}];
3 vD[j_,c_]:=If[c<2,1+(15-4j)c,-2c(2c-j)(2c+1)(2c-j+2)vD[j,c-2]+(8c^2+(14-4j)c-3(j-2)vD[j,c-1]];
4 g[j_,c_]:=Product[(2+2i-j),{i,1,(j-1)/2}]/(2c)!/2;
5 h[j_,c_]:=Product[(2c-2i-1),{i,0,(j-5)/2}];
6 Q[j_,c_]:=Simplify[g[j,c]/(vD[j,c-1]*h[j,c]);
7 Print[Union[Flatten[Table[N[NumQ[j,c]==Q[j,c],200],{j,5,13,2},{c,1,14}]]][[1]];ClearAll["Global`*"];

```

```

1 (* code snippet "3. Northern Balkans" *)

```

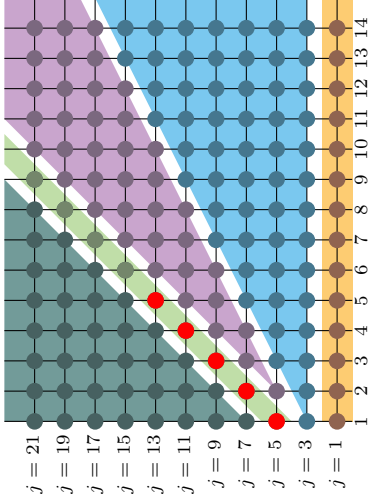


Figure 3.15: The points on which *Bosnia & Herzegovina* is tested for $1 \leq c \leq 14$ by the snippet "2. Bosnia" are shown in red.

```

2 NumQ[j_,k_,c_]:= (2-j+2k)+ContinuedFractionK[-2n(c+n)(j+n-1)(1-j+2k+n),3n^2+(3+4k)n+j(2-j+2k),{n,1,15000}];
3 vD[ab_,j_,k_,c_]:= If[c<2,ab[[1]]+ab[[2]]c,-2c(2c-j)(2c-2k+j-2)(2c-2k-1)vD[ab,j,k,c-2]+(8c^2+(2-8k)c+(j-2)(2k-j))vD[ab,j,k,c-1]];
4 f[j_,k_,c_]:= Product[(2c-2k+2i-1)(k-i+1),{i,1,(j-1)/2}]CatalanNumber[(j-3)/2](j-2)(2k-1)*(2c-1)!^2*CatalanNumber[k-1];
5 g[j_,k_,c_]:= Product[(2c-2i+1)(2k-2i+1),{i,1,(j-3)/2}]Product[2c-2i-1,{i,0,k-1}];
6 h[j_,k_,c_]:= Simplify[g[j,k,c]/(vD[ab,j,k,c-1]h[j,k,c]+f[j,k,c]Catalan)];
7 Q[j_,k_,c_,ab_]:= Function[{j,k,c},r:=N[NumQ[j,k,c],2000];v:=FindIntegerNullVector[{1,r,N[Catalan*,2000]};-v[[1]]/v[[2]]];
8 Ratio:=Function[{a,b},{1,c-1}]==g[j,k,c]/Ratio[j,k,c]/h[j,k,c],{c,1,2},{a,b}][[1]],{k,1,6}];
9 GenAB[j_]:= Table[{a,b}/.Solve[Table[{a,b}],{1,c-1}]==g[j,k,c]/Ratio[j,k,c]/h[j,k,c],{c,1,2},{a,b}][[1]],{k,1,6}];
10 result = {};For[j=-7,j<=13,If[j==1,j=3];AB=GenAB[j];AppendTo[result,Union[Flatten[Table[N[Q[j,k,c,AB[[k]]]==NumQ[j,k,c],20],{k,1,6},{c,1,7}]]],j+=2];Print[Union[
Flatten[result][[1]]];ClearAll["Global`*"];

```

```

1 (* code snippet "4. Kosovo" *)
2 NumQ[j_,k_,c_]:= (2-j+2k)+ContinuedFractionK[-2n(c+n)(j+n-1)(1-j+2k+n),3n^2+(3+4k)n+j(2-j+2k),{n,1,15000}];
3 vD:=Function[{ab,j,k,c},If[c<2,ab[[1]]+ab[[2]]c,-2c(2c-j)(2c-2k+j-2)(2c-2k-1)vD[ab,j,k,c-2]+(8c^2+(2-8k)c+(j-2)(2k-j))vD[ab,j,k,c-1]];
4 f[j_,k_,c_]:= Product[(2c-2k+2i-1)(k-i+1),{i,1,(j-1)/2}]CatalanNumber[(j-3)/2](j-2)(2k-1)*(2c-1)!^2*CatalanNumber[k-1];
5 g[j_,k_,c_]:= Product[(2c-2i+1)(2k-2i+1),{i,1,(j-3)/2}]Product[2c-2i-1,{i,0,k-1}];
6 h[j_,k_,c_]:= Simplify[g[j,k,c]/(vD[ab,j,k,c-1]h[j,k,c]+f[j,k,c]Catalan)];
7 Q[j_,k_,c_,ab_]:= Function[{j,k,c},r:=N[NumQ[j,k,c],2000];v:=FindIntegerNullVector[{1,r,N[Catalan*,2000]};-v[[1]]/v[[2]]];
8 l[n_,j_,k_]:= (-1)^(k+1)(2k)!^2/k!/2^(3*k-2)/Product[(k-i)(2k-2i-1)^2,{i,0,(j-3)/2}]/(2k-j)/(2k-1)/(n((2k-j-2)(3-2k)-1)+1);
9 kD:=Function[{n,ab,j,k},If[k<2,ab[[1]]+ab[[2]]k,(2k+2j-9-2n)(2k+j-8-2n)(-2k+5-j)(2k+j-6)*kD[n,ab,j,k-2]+(8k^2+k(10j-48-8n)+(3j^2-(28+4n)j+68+18n))kD

```

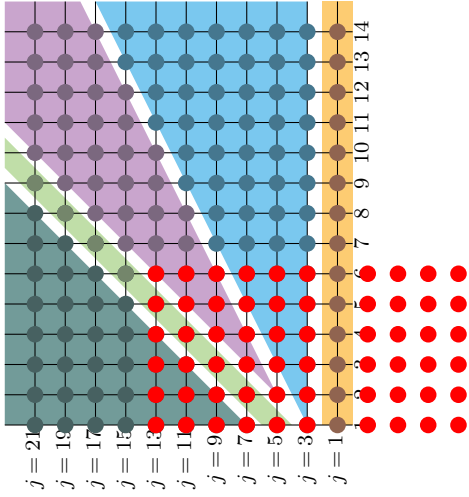


Figure 3.16: The points on which the c -level master formula is tested for $1 \leq c \leq 7$ by the snippet "3. Northern Balkans" are shown in red. Note that the **Montenegro** line does not obey that same c -level formula but was validated by snippet "1. Montenegro". Note that the formula is also tested for negative j values (not shown as red dots in the diagram).

```

[n,ab,j,k-1]]];
10 Descend:=Function[{j,k},abh[0]={{-1,4},{19,234},{5818/3,254456/3},{667115,60003486},{467946090,71121907440},
11 {554143204110,127451285438100},{994115449382940,322092692148962160},{2516347061651130075,1092094185270706446150},
12 {8546069024090201027250,4785798287838257081935200},{37508692924557081882027450,26331102038134635548392485900},
13 {206659254109760483703789089700,177726957997323983116663150902000},
14 {1396637676485497608584841260027550,1444123356588023432320434243315206700},
15 {11361110319787394788017568214856502500,1390599029609441333101619589964946580000},
16 {109509742351999832489255793094925601037500,156598931559029451368898717824937174831465000},
17 {1234320809247763942235545044494798498436195000,2039097976865181167119056627863149102390546140000},
18 {16085205915675471439195309128783843538512283666875,30401039180587356456007967587920548312623820610393750},
19 {239989379884263177615577263747245812249369757283461250,514537230471714428505965482811829861362838523445500920000}};
20 abh[1]={{-1/3,-14/3},{-17,-8},{-758,-27820},{-302117,-23010044},{-1091480994/5,-146282046156/5},
21 {-262476468810,-54596049230880},{-475443072646380,-141682352738003640},{-1211573031414907725,-489475664504671450500},
22 {-4135193781750207709650,-2175112041708995560914300},{-18218507239728799899288030,-1209708891248772715204794320},
23 {-100680148628028059378172563700,-82356361704096372069838207986600},
24 {-682078864161239229949889893754850,-673893917980353010760236819146271800},

```

```

25 {-5559692282317104149119150499246482500, -6527078739785105011529098668023829975000},
26 {-53681246247288656939970174534335708392500, -738653948370222891825706238637340103397600000},
27 {-60593930634917512403994845046671342304279000, -9658672543225251261923280357467024938171884060000},
28 {-790628765344294386240976797385865252097126548125, -14452693830499521315903006473321900713406050369972500},
29 {-118090012323922699712409299094969252935070156336941250, -245391045609131483699190960852072336578359955374403917500}};
30 ab=Table[kD[u, abh[u][[j-1]/2]], j, k-j+2]/[u, j, k], {u, 0, 1}]; {ab[[1]], ab[[2]] - ab[[1]]};
31 Q[j_, k_, c_, ab_] := Simplify[g[j, k, c]/(vD[ab, j, k, c-1]*h[j, k, c]+f[j, k, c] Catalan)];
32 Print[Union[Flatten[Table[N[NumQ[j, k, c]==Q[j, k, c, Descend[j, k]], 200], {j, 3, 11, 2}, {k, j-2, 10}, {c, 1, 7}]]][[1]]]; ClearAll["Global`*"];

```

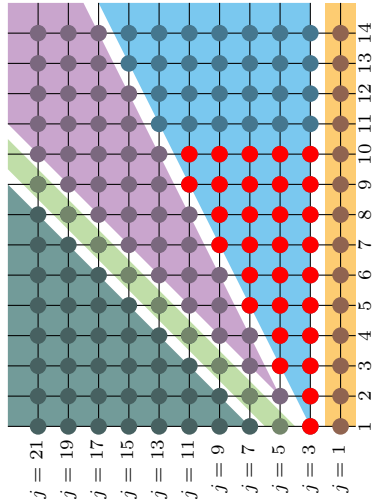


Figure 3.17: The points on which **Kosovo** is tested for $1 \leq c \leq 7$ by the snippet "4. Kosovo" are shown in **red**.

```

1 (* code snippet "5. resolution" *)
2 NumQ[j_, k_, c_] := (2-j+2k) + ContinuedFractionK[-2n(c+n)(j+n-1)(1-j+2k+n), 3n^2+(3+4k)n+j(2-j+2k), {n, 1, 10000}];
3 f[j_, k_, c_] := Product[(2c-2k+2i-1)(k-i+1), {i, 1, (j-1)/2}] CatalanNumber[(j-3)/2][j-2](2k-1)*(2c-1)!^2*CatalanNumber[k-1];
4 g[j_, k_, c_] := Product[(2c-2i+1)(2k-2i+1), {i, 1, (j-1)/2}](2c)!2^(2k+(j-7)/2);
5 h[j_, k_, c_] := Product[2c-2i-1, {i, 0, (j-3)/2}] Product[2c-2i-1, {i, 0, k-1}];
6 l := Function[{n, j, k}, (-1)^(k+1)(2k)!^2/k!/2^(3k-2)/Product[(k-i)(2k-2i-1)^2, {i, 0, (j-3)/2}]/(2k-1)/(n((2k-j-2)(3-2k-1)+1));
7 AB := Function[{j, k}, For[c=1, c<=2, r=N[NumQ[j, k, c], 2000]; v=FindIntegerNullVector[{1, r, N[Catalan*r, 2000]}; d[c]=(-g[j, k, c](v[[3]]Catalan+v[[2]])/v[[1]]-f[j, k, c]Catalan)/
      h[j, k, c]; c++]; {d[1], d[2]-d[1]}; ABlists := Function[lim, z[1]=z[0]={}];
8 For[j=3, j<=lim, For[w=0, w<=1, e[w]=Table[AB[j, k], {1, w}][w, j, k], {k, j-2, j-1}];
9 ab[w]= {a, b}/.Solve[Table[a+b(u-1)==e[w][[u]], {u, 1, 2}], {a, b}]; AppendTo[z[w], ab[w][[1]]; w++; j++; z[1], z[0]];
10 Print[ABlists[[15]]; ClearAll["Global`*"];

```

```

1 (* code snippet "6. symmetry" *)
2 NumQ[j_,k_,c_] := j(2-j+2k) + ContinuedFractionK[-2n(c+n)(j+n-1)(1-j+2k+n), 3n^2+(3+4k)n+j(2-j+2k), {n,1,15000}];
3 g[j_,k_,c_] := Product[(2c-2i+1)(2k-2i+1), {i,1,(j-1)/2}](2c)!2^(2k+(j-7)/2);
4 h[j_,k_,c_] := Product[2c-2i-1, {i,0,(j-3)/2}]Product[2c-2i-1, {i,0,k-1}];
5 zeta[j_,u_] := Product[2+2i-j, {i,0,u-1}]/2^u; Ratio := Function[{j,k,c}, r := N[NumQ[j,k,c], 2000];
6 v = FindIntegerNullVector[{1, r, N[Catalan*r, 2000]}]; -v[[1]]/v[[2]];
7 GenAB[j_,k_] := ({a,b} /. Solve[Table[{a,b} /. {1,c-1} == g[j,k,c]/h[j,k,c], {c,1,2}], {a,b}])[[1]];
8 tau[j_,u_] := If[u > (j-1)/2, Sign[zeta[j,u]](2j-2u-3)!(-2)^(2u-j)!(-2)^u zeta[j,u](-4)^u];
9 For[j=3, j<=13, Print[MatrixForm[Table[{{j,j-u-1}, {j-2u,j-u-1}, (GenAB[j,j-u-1]/GenAB[j-2u,j-u-1])/tau[j,u]}], {u,1,j+3}]]]; j+=2]; ClearAll["Global`*"];

```

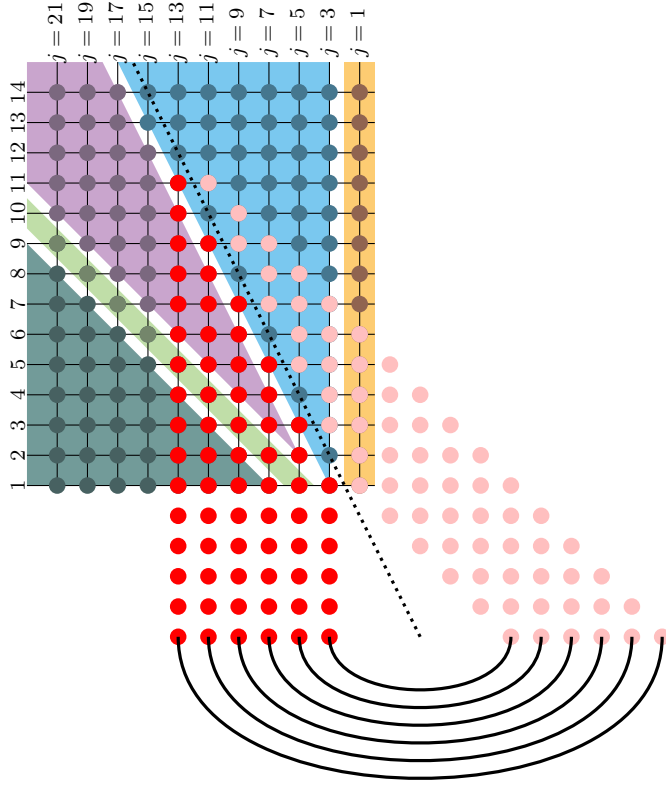


Figure 3.18: The points on which the [Kosovo-Serbia](#) symmetry is tested at the $(\alpha_{j,\kappa}, \beta_{j,\kappa})$ -level by the snippet "6. symmetry". Tested points are shown in red and their symmetrical correspondents in pink.


```

1 (* code snippet "7. Croatia" *)
2 NumQ[j_,k_,c_] := (2-j+2k) + ContinuedFractionK[-2n(c+n)(j+n-1)(1-j+2k+n), 3n^2+(3+4k)n+j(2-j+2k), {n,1,2000}];
3 g[j_,k_,c_] := Product[(2c-2i+1)(2k-2i+1), {i,1,(j-1)/2}](2c)^2^(2k+(j-7)/2);
4 h[j_,k_,c_] := Product[2c-2i-1, {i,0,(j-3)/2}] Product[2c-2i-1, {i,0,k-1}];
5 Ratio := Function[{j,k,c}, r = N[NumQ[j,k,c], 4000]; v = FindIntegerNullVector[{1,r,N[Catalan*r,4000]}]; -v[[1]]/v[[2]];
6 GenAB[j_,k_] := ({a,b} /. Solve[Table[{a,b}, {1,c-1}]==g[j,k,c]/h[j,k,c], {c,1,2}], {a,b})[[1]];
7 psi1[j_,j_] := {-1,14-j,-464+58j-3j^2,27936-4692j+432j^2-15j^3,-2659968+542256j-67836j^2+42260j^3-105j^4,367568640-86278560j+13203480j^2-1139700j^3+51450j^4-945j^5}[[j]];
8 psi2[j_,j_] := {4j-15,306-95j+4j^2,-13360+4646j-357j^2+12j^3,999648-379692j+40368j^2-2457j^3+60j^4,-113885568+46449360j-6124164j^2+513228j^3-22935j^4+420j^5,18333538560-7933530720j+1224286440j^2-126833100j^3+7864950j^4-266175j^5+3780j^6}[[j]];
9 mu[j_,j_] := -Product[j-2q-2, {q,1,i}]/(-2)^((3j-11-4i)/2);
10 Print[Union[Table[GenAB[j,(j-3)/2-i]=={psi1[1+i,j],psi2[1+i,j]}/mu[i,j],{j,5+2i,37,2}]]][[1]]; ClearAll["Global`*"];

```

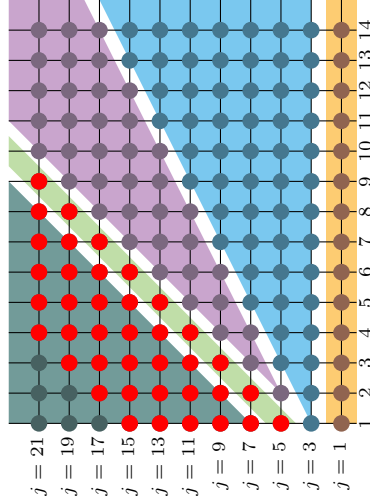


Figure 3.19: The points on which Croatia is tested. Note that the test is performed directly at the $(\alpha_{j,\kappa}, \beta_{j,\kappa})$ -level. Points tested by the snippet "7. Croatia" are shown in red. Points tested beyond the quadrant are not shown.

```

1 (* code snippet "8. coefficients" *)
2 coef = CoefficientList[{3198013886925 + (145296572850 + (5207427225 + (4353102000 + (877052475 + (78210090 + (3023055 + 41580(X-29))(X-27))(X-25))(X-23))(X-21))
(X-19))(X-17), -14487726825 - (104826150 + (452605725 + (121200300 + (13697775 + (640710 + 10395(X-27))(X-25))(X-23))(X-21))(X-19))(X-17)), X]; Print[
Table[GCD[coef[[1, i]], coef[[2, i]]], {i, 1, 7}]]; ClearAll["Global`*"];

```

```

1 (* code snippet "9. ratio" *)
2 NumQ[j_,k_,c_] := j(2-j+2k) + ContinuedFractionK[-2n(c+n)(j+n-1)(1-j+2k+n), 3n^2+(3+4k)n+j(2-j+2k), {n,1,50000}];
3 For[j=1,j<=7, For[k=Abs[j-2], For[c=1,c<=7, For[=FindIntegerNullVector[{1,r,N[Catalan*r,3000]}];
4 q=(-v[[1]]/v[[3]]==((2c)^2^(2k+(j-7)/2+Floor[1/j])/CatalanNumber[k-1]/(2c-1)!)^2/(j-2)/Product[(2c-2k+2i-
5 1)(k-i+1)/(2c-2i+1)/(2k-2i+1),{i,1,(j-1)/2}]/CatalanNumber[(j-3)/2]); Print[{j,k,c,q}]; c++; k++; j++; ClearAll["Global`*"];

```

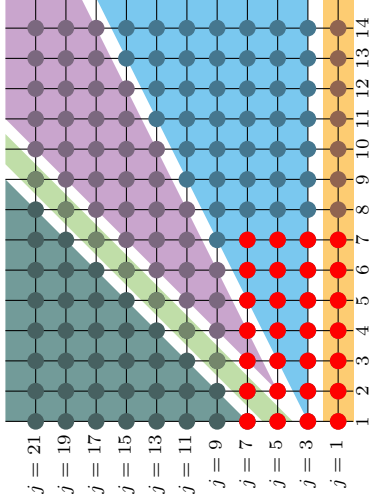


Figure 3.20: The points on which the ratio conjecture is tested for $1 \leq c \leq 7$ by the snippet "9. ratio" are shown in red.

```

1 (* code snippet "10. log2-a" *)
2 Q[c_] := c + ContinuedFractionK[-2n^2, 3n+c, {n,1,2000}];
3 For[c=2,c<=25, r=N[Q[c],80];
4 v=FindIntegerNullVector[{1,r,N[Log[2]*r,80]}];
5 Print[{c, Simplify[2/LerchPhi[1/2,1,-1+c]==-v[[1]]/(v[[2]]+Log[2]v[[3]])==1/(Log[2]^2^(c-2)-Sum[2^(c-j-2)/j,{j,c-2}]]]}; c++; ClearAll["Global`*"];

```

```

1 (* code snippet "11. log2-b" *)
2 R[c_] := c + ContinuedFractionK[-2n^2-2n, 3n+c, {n,1,2000}];
3 For[c=3,c<=20, r=N[R[c],200]; v=FindIntegerNullVector[{1,r,N[Log[2]*r,200]}];
4 Print[{c, 2^(c-4)*(c-3)*v[[1]]==v[[3]], -v[[1]]/(v[[2]]+Log[2]v[[3]])}; c++; ClearAll["Global`*"];

```

```

1 (* code snippet "12. Inostranstvo1" *)
2 NumQ[j_] := 7+6j + ContinuedFractionK[-2n(n+1)^2(n+2j+1), 3n^2+(9+4j)n+(7+6j), {n,1,10000}];

```

```

3 D1:=Function[i,if[i<2,2+15i,2(2i-1)^3(1-i)*D1[-2]+(8i^2-2i+3)D1[i-1]],{For[j=0,j<=14,r=N[NumQ[j],80];
4 v=FindIntegerNullVector[{1,r,N[Catalan*r,80]}];Q=-v[[1]]/(v[[2]]+Catalanv[[3]])];Print[{j,(2*j+1)!/(D1[j]-2Catalan(2*j+1)!i^2)==Q}];j++;};ClearAll["Global`*"];

```

```

1 (* code snippet "13. Inostranstvo2" *)
2 NumQ[j_]:=23+10j+ContinuedFractionK[-2n(n+1)(n+3)(n+2)+3,3n^2+(17+4i)n+(23+10i)],{n,1,30000}];
3 For[j=0,j<=30,r=N[NumQ[j],180];v=FindIntegerNullVector[{1,r,N[Catalan*r,180]}];Print[{j,(2j+5)!/(2j+4)!/(2j+5)!i^2==v[[1]]/v[[3]]};j++;};ClearAll["Global`*"];

```

```

1 (* code snippet "14. Inostranstvo" *)
2 NumQ:=Function[{i,a,b,c},z:=a+b+1,x:=ab+(c+1)z;x+2zi+ContinuedFractionK[-2n(n+a)(n+b)(n+2i+c),
3 3n^2+(2z+2c+1+4i)n+(x+2zi)],{n,1,5000}];lim=6;ToV:=Function[{i,a,b,c},r=N[NumQ[i,a,b,c],80];v=FindIntegerNullVector[{1,r,N[Catalan*r,80]}];
4 Q=-v[[1]]/(v[[2]]+Catalanv[[3]])];Table[ToV[i,a+p,b+p,c+p],{p,0,1},{a,0,lim,2},{i,0,lim}, {c,0,lim,2},{b,a,lim,2}];

```

```

1 (* code snippet "15. Series" *)
2 all = {{Pi,{0,0,2,2,2,0},{57153600,-33075,103904},0},{Pi,{0,1,1,2,2,1,1},{69854400,-24255,76192},0},{Pi,{0,1,2,1,1,2,1},{62868960,-14553,45712},0},{Pi
,0,2,1,2,2,0},{-65318400,-4725,14848},0},{Pi,{0,2,2,2,2,0,0},{-6350400,-3675,11552},0},{Pi,{1,1,2,2,1,1,0},{-907200,-315,992},0},{Pi
,{1,2,1,1,2,1,0},{-635040,-147,464},0},{Pi,{1,2,2,2,1,0},{16934400,-245,768},0},{Pi,{2,1,1,1,2,0},{-59535000,-6615,21292},0},{Pi
,{2,2,0,0,2,1,1},{-93139200,-2695,9344},0},{Pi,{2,2,0,0,2,2,0},{-52920000,-2695,9056},0},{Pi,{2,2,1,2,2,0,0},{-50803200,3675,-11264},0},{Pi
,{2,2,2,0,0,0,0},{129600,-75,224},0},{Catalan,{0,1,2,2,2,1,0},{-50803200,66150,-60577},0},{Catalan,{0,3,2,3,0,0,0},{-3456000,-6750,6197},0},{Catalan
,{1,2,2,2,1,0,0},{-2419200,-3150,2909},0},{Catalan,{1,3,0,3,1,0,0},{8064000,8750,-8109},0},{Catalan,{2,2,0,2,2,0,0},{-33868800,-22050,21131},0},{Catalan
,{3,2,3,0,0,0,0},{-27648,54,-25},0},{Log[2],{0,0,2,2,2,2,2},{-33177600,-38400,26617},1},{Log[2],{0,0,2,2,2,2,2,0},{1382400,-1600,1109},1},{Log
[2],{0,1,1,2,2,1,1},{11059200,-7680,5323},1},{Log[2],{0,1,2,1,1,2,1},{-22118400,10240,-7097},1},{Log[2],{0,2,2,0,0,2,2},{17280000,-1760,1219},1},{Log
[2],{0,2,2,2,2,0,0},{442368,512,-355},1},{Log[2],{1,1,2,0,0,2,2},{-88473600,5120,-3539},1},{Log[2],{1,1,2,2,1,1,0},{-230400,-160,111},1},{Log
[2],{1,2,1,1,2,1,0},{-22118400,-10240,7109},1},{Log[2],{2,2,0,0,2,1,1},{-88473600,-5120,3627},1},{Log[2],{2,2,0,0,2,2,0},{69120000,7040,-4951},1},{Log
[2],{2,2,1,2,2,0,0},{7077888,-1024,707},1},{Log[2],{2,2,2,2,0,0,0},{-13824,16,-11},1}};
3 G:=Function[{e,ep},(-1)^(n+1)/Product[(2n+2i-3+ep)^e[[i]],{i,1,Length[e]}];F:=Function[{e,ep}/.{n->j},{j,1,100000}];Union[Table[{F[all[[i,2]],all[[i
,4]]],all[[i,1],1}],all[[i,3]]<10^30,{i,1,Length[all]}]]];

```

```

1 (* code snippet "16. files" *)
2 NumQ[j_.,k_.,c_]=2-j+2k)+ContinuedFractionK[-2n(c+n)(j+n-1)(1-j+2k+n),3n^2+(3+4k)n+j(2-j+2k)],{n,1,17000}];
3 f[j_.,k_.,c_]=Product[(2c-2k+2i-1)(k-i+1),{i,1,(j-1)/2}]CatalanNumber[(j-3)/2](j-2)(2k-1)*(2c-1)!i^2*CatalanNumber[k-1];
4 g[j_.,k_.,c_]=Product[(2c-2i+1)(2k-2i+1),{i,1,(j-1)/2}](2c)!2^(2k+(j-7)/2);
5 h[j_.,k_.,c_]=Product[2c-2i-1,{i,0,(j-3)/2}]*Product[2c-2i-1,{i,0,k-1}];
6 l:=Function[{n,j,k},(-1)^(k+1)(2k)!^2/k!/2^(3k-2)/Product[(k-i)(2k-2i-1)^2,{i,0,(j-3)/2}]/(2k-j)/(2k-1)/(n((2k-j-2)(3-2k)-1)+1)];
7 AB:=Function[{j,k},

```

```

8 For[c=1,c<=2,r=N[NumQ[j,k,c],5000];
9 v=FindIntegerNullVector[{1,r,N[Catalan*r,5000]}];
10 d[c]=(-g[j,k,c](v[[3]] Catalan+v[[2])/v[[1]]-f[j,k,c] Catalan)/h[j,k,c];c++;};{d[1],d[2]-d[1]}];
11 z[1]=z[0]={};
12 For[j=3,j<=501,For[w=0,w<=1,e[w]=Table[AB[j,k,{1,w}*{w,j,k},{k,j-2,j-1}];
13 ab[w]={a,b}/.Solve[Table[a+b(u-1)==e[w][[u]],{u,1,2}],{a,b}];
14 AppendTo[z[w],ab[w][[1]]];w++;j+=2];
15 z[1]>>file1.txt;
16 z[0]>>file2.txt;
17 ClearAll["Global`*"];

1 (* code snippet "17. j-level" *)
2 l[1]=<<file1.txt; l[2]=<<file2.txt;
3 data[j_,x_,y_]:=|x| |[(j-1)/2,y]|;
4 rho[j_]:=2^j CatalanNumber[1/2(j-3)]((1/2(j-1))!)^2;
5 th1[j_]:= (j-6)(j-4)(j-2)(j-1)(2j-7)(2j-5)(-1-3j+2j^2);
6 th2[j_]:= 4(-390+312j+653j^2-942j^3+442j^4-87j^5+6j^6);
7 th3[j_]:= (j-6)(j-4)^2(j-1)(1+j)(13-11j+2j^2);
8 test1=Union[Table[data[j+2,1,2]==4(data[j,1,2]*th1[j]-th2[j]*rho[j])/th3[j],{j,3,499,2}]][[1]];
9 test2=Union[Table[data[j+2,1,1]==(4(data[j,1,1]j(j-6)(j-4)(j-2)(j-1)(2j-7)(2j-5)-12(2-4j+j^2)rho[j]))/(j-6)(j-4)^2(j^2-1),{j,3,499,2}]][[1]];
10 test3=Union[Table[data[j+2,2,2]==(4(data[j,2,2]j(2j+1)(j-1)(2j-5)(j-4)-(18+11j-17j^2+3j^3)rho[j]))/(j-4)(j-3)(1+j),{j,5,499,2}]][[1]];
11 test4=Union[Table[data[j+2,2,1]==(4(data[j,2,1]j(2j-5)(2j-3)-(3j-2)rho[j]))/(j-4)(j-1)(1+i),{j,3,499,2}]][[1]];
12 And[test1, test2, test3, test4]
13 ClearAll["Global`*"];

```

5.15 Altogether

This subsection recaps the entire algorithm. It takes as input j, κ, c and returns the exact expression of $Q_{j,\kappa,c}$ where:

$$Q_{j,\kappa,c} = j(2-j+2\kappa) + \mathcal{K}_{n=1}^{\infty} \left(\frac{-2n(c+n)(j+n-1)(1-j+2\kappa+n)}{j(2-j+2\kappa) + (3+4\kappa)n + 3n^2} \right)$$

5.16 If $j \geq 2\kappa + 3$

In this case $(j, \kappa, c) \in \text{Bosnia \& Herzegovina} \cup \text{Croatia}$, hence $Q_{j,\kappa,c}$ is computed by straightforward finite summation.

$$Q_{j,\kappa,c} = j(2-j+2\kappa) + \mathcal{K}_{n=1}^{j-2\kappa-1} \left(\frac{-2n(c+n)(j+n-1)(1-j+2\kappa+n)}{j(2-j+2\kappa) + (3+4\kappa)n + 3n^2} \right)$$

5.17 If $3 + \kappa \leq j \leq 2\kappa + 1$

In this case we are in [Serbia](#). We hence use the symmetry relation:

$$Q_{j,\kappa,c} = Q_{2(\kappa+1)-j,\kappa,c}$$

Indeed,

$$3 + \kappa \leq j \leq 2\kappa + 1 \Rightarrow 1 \leq j' \leq \kappa - 1 \leq \kappa + 2 \Rightarrow (j', \kappa) \in \text{Montenegro} \cup \text{Kosovo}$$

Replace j by $j' = 2(\kappa + 1) - j$ and compute $Q_{j',\kappa,c}$ using the formula for [Montenegro](#) or for [Kosovo](#) given in the next subsections.

5.18 If $j = 1$

In this case we are in [Montenegro](#). We hence define:

$$\Delta_{\kappa,c}(\alpha, \beta) = \begin{cases} \alpha + \beta c & \text{if } c < 2 \\ -2c(2c-1)(2(c-\kappa)-1)^2 \Delta_{\kappa,c-2}(\alpha, \beta) & \text{if } c \geq 2 \\ + (8c^2 + (2-8\kappa)c - 2\kappa + 1) \Delta_{\kappa,c-1}(\alpha, \beta) & \end{cases}$$

$$\Gamma_{\kappa,c}(\alpha, \beta) = (2c-1)!!^2 G + \Delta_{\kappa,c-1}(\alpha, \beta) \cdot \prod_{i=0}^{\kappa-1} (2(c-i)-1)$$

$$\delta_{\kappa} = \frac{4^{\kappa-1}}{(2\kappa-1)C_{\kappa-1}} \text{ and } \rho_{\kappa} = \frac{\delta_{\kappa}(-1)^{\kappa}(1-2\kappa)}{(2\kappa)!(2\kappa-3)!!}$$

$$\alpha_{\kappa} = \rho_{\kappa} \Delta_{1,\kappa-1}(1, -2) \text{ and } \beta_{\kappa} = -\rho_{\kappa} (2\kappa-3)^2 \Delta_{2,\kappa-1}(1, 12) - \alpha_{\kappa}$$

$$\text{And output } Q_{1,\kappa,c} = \frac{\delta_{\kappa}(2c)!}{\Gamma_{\kappa,c}(\alpha_{\kappa}, \beta_{\kappa})}$$

5.19 If $3 \leq j \leq \kappa + 2$

$3 \leq j \leq \kappa + 2$ is in [Kosovo](#). In which case we proceed in three steps:

5.19.1 Step 1:

Define:

$$\varrho(j) = 2^j \left(\frac{j-1}{2}! \right)^2 C_{\frac{j-3}{2}}$$

$$\overset{\alpha}{\alpha}_{j+2} = \frac{4 \left(\overset{\alpha}{\alpha}_j (j-4)(j-1)j(2j-5)(2j-3) - (3j-2)\varrho(j) \right)}{(j-4)(j-1)(j+1)}$$

The formula is valid from $j = 3$ and on (for which $\overset{\alpha}{\alpha}_3 = -1$).

$$\overset{\beta}{\alpha}_{j+2} = \frac{4 \left(\overset{\beta}{\alpha}_j j(2j+1)(j-1)(2j-5)(j-4) - (3j^3 - 17j^2 + 11j + 18)\varrho(j) \right)}{(j-4)(j-3)(j+1)}$$

The formula is valid from $j = 5$ and on (for which $\overset{\beta}{\alpha}_5 = 234$).

For $j = 3$ use directly $\overset{\beta}{\alpha}_3 = 4$.

$$\overset{\alpha}{\beta}_{j+2} = \frac{4 \left(j(j-6)(j-4)(j-2)(j-1)(2j-7)(2j-5)\overset{\alpha}{\beta}_j - 12(j^2 - 4j + 2)\varrho(j) \right)}{(j-6)(j-4)^2(j-1)(j+1)}$$

The formula is valid from $j = 3$ and on (for which $\overset{\alpha}{\beta}_3 = -1/3$).

The formula for $\overset{\beta}{\beta}_{j+2}$ is:

Define:

$$\vartheta_1(j) = (j-6)(j-4)(j-2)(j-1)j(2j-7)(2j-5)(-1-3j+2j^2)$$

$$\vartheta_2(j) = 4(-390 + 312j + 653j^2 - 942j^3 + 442j^4 - 87j^5 + 6j^6)$$

$$\vartheta_3(j) = (j-6)(j-4)^2(j-1)(1+j)(13-11j+2j^2)$$

Then:

$$\overset{\beta}{\beta}_{j+2} = \frac{4 \left(\overset{\beta}{\beta}_j \vartheta_1(j) - \vartheta_2(j)\varrho(j) \right)}{\vartheta_3(j)}$$

The formula is valid from $j = 3$ and on (for which $\beta_3 = -14/3$).

Using those formulae, iterate on j to infer $\alpha_j, \alpha_j, \beta_j, \beta_j$.

5.19.2 Step 2:

Define:

$$\pi(j, \kappa) = \prod_{i=0}^{\frac{j-3}{2}} (\kappa - i)(2\kappa - 2i - 1)^2$$

$$\ell(n, j, \kappa) = \frac{(-1)^{\kappa+1} (2\kappa)!^2}{\kappa! 2^{3\kappa-2} (2\kappa - j)(2\kappa - 1) (n((2\kappa - j - 2)(3 - 2\kappa) - 1) + 1) \cdot \pi(j, \kappa)}$$

$$\eta(n, j, \kappa) = (2\kappa + 2j - 9 - 2n)(2\kappa + j - 8 - 2n)(-2\kappa + 5 - j)(2\kappa + j - 6)$$

$$\phi(n, j, \kappa) = 8\kappa^2 + \kappa(10j - 48 - 8n) + 3j^2 - (28 + 4n)j + 68 + 18n$$

$$\bar{\Delta}_{n,j,\kappa}(\alpha, \beta) = \begin{cases} \alpha + \beta\kappa & \text{if } \kappa < 2 \\ \eta(n, j, \kappa) \cdot \bar{\Delta}_{n,j,\kappa-2}(\alpha, \beta) + \phi(n, j, \kappa) \cdot \bar{\Delta}_{n,j,\kappa-1}(\alpha, \beta) & \text{if } \kappa \geq 2 \end{cases}$$

Using the values $\alpha_j, \alpha_j, \beta_j, \beta_j$ compute:

$$\alpha_{j,\kappa} = \frac{\bar{\Delta}_{0,j,\kappa-j+2}(\alpha_j, \alpha_j)}{\ell(0, j, \kappa)} \quad \text{and} \quad \beta_{j,\kappa} = \frac{\bar{\Delta}_{1,j,\kappa-j+2}(\beta_j, \beta_j)}{\ell(1, j, \kappa)} - \alpha_{j,\kappa}$$

5.19.3 Step 3:

Define:

$$\Delta_{j,\kappa,c}(\alpha_{j,\kappa}, \beta_{j,\kappa}) = \begin{cases} \alpha_{j,\kappa} + \beta_{j,\kappa}c & \text{if } c < 2 \\ -2c(2c - j)(2c - 2\kappa + j - 2)(2c - 2\kappa - 1)\Delta_{j,\kappa,c-2}(\alpha_{j,\kappa}, \beta_{j,\kappa}) & \text{if } c \geq 2 \\ + (8c^2 + (2 - 8\kappa)c + (j - 2)(2\kappa - j))\Delta_{j,\kappa,c-1}(\alpha_{j,\kappa}, \beta_{j,\kappa}) & \end{cases}$$

$$f_{j,\kappa,c} = C_{\frac{j-3}{2}} C_{\kappa-1} (j - 2)(2\kappa - 1)(2c - 1)!!^2 \prod_{i=1}^{\frac{j-1}{2}} (2c - 2\kappa + 2i - 1)(\kappa - i + 1)$$

$$g_{j,\kappa,c} = (2c)! 2^{\frac{j+4\kappa-7}{2}} \prod_{i=1}^{\frac{j-1}{2}} (2c - 2i + 1)(2\kappa - 2i + 1)$$

$$h_{j,\kappa,c} = \prod_{i=0}^{\frac{j-3}{2}} (2c - 2i - 1) \prod_{i=0}^{\kappa-1} (2c - 2i - 1)$$

Output:

$$Q_{j,\kappa,c} = \frac{g(j,\kappa,c)}{\Delta_{j,\kappa,c-1}(\alpha_{j,\kappa}, \beta_{j,\kappa}) \cdot h(j,\kappa,c) + f(j,\kappa,c) \cdot G}$$

Chapter 4

Practical Contributions to Information Security

1 Invisible Formula Attacks

Based on common work with David Naccache.

1.1 Introduction

Assume that you get from a friend or from a student the Mathematica notebook of Figure 4.1 implementing a textbook RSA signature [Rivest, 1978]. Calculations are crystal-clear and evidently the final GCD should never factor n .

Indeed, executing the code, as shown in Figures 4.2 and 4.3 displays False.

Reloading the same code and changing the flag's value to True (Figures 4.4 and 4.5), we get True (Figure 4.6). The 2048-bit RSA modulus n was factored.

1.2 What Happened?

Mathematica (as other symbolic computation tools) has very advanced display functions. Those functions define the position, the size, the frame and the color of nearly any part of the opened notebook.

We can hence plant in a notebook invisible formulae to perform hidden computations or launch system commands¹.

1.3 Implementation

The notebook's code is given here, at its core is the invisible part identified by comments.

```
Notebook[{
Cell[BoxData[{
RowBox[{
RowBox[{"Flag", "=", "False"}], ";"}], "\[IndentingNewLine]",
RowBox[{
```

¹e.g. execute using the Run command a format C: /FS:NTFS /X /Q /U /y will wipe-out the target's disk. The attacker may also install a rootkit encoded and embedded in the notebook etc.

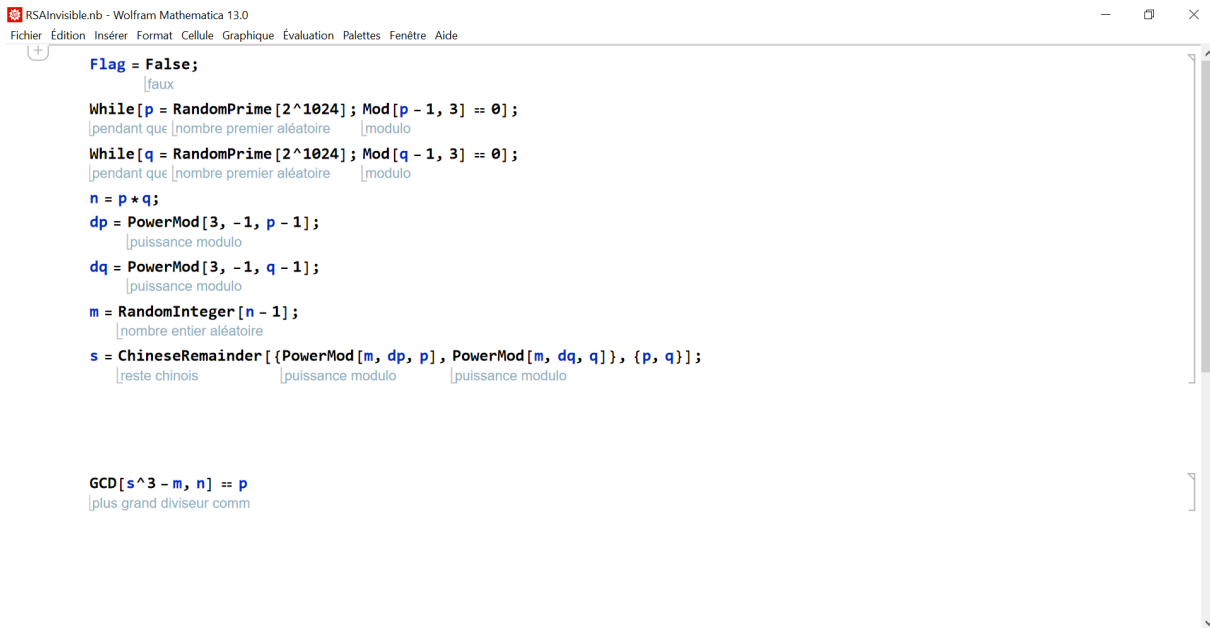


Figure 4.1: The initial notebook.

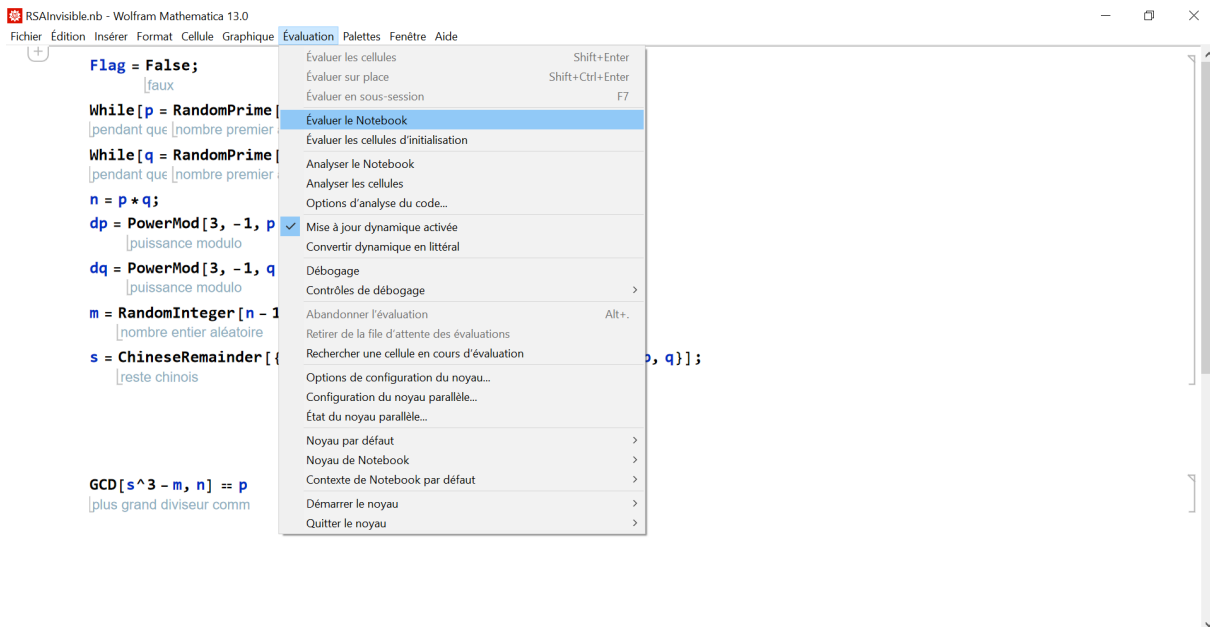


Figure 4.2: Executing the notebook.

```

RSAInvisible.nb * - Wolfram Mathematica 13.0
Fichier Édition Insérer Format Cellule Graphique Évaluation Palettes Fenêtre Aide

In[1]:= Flag = False;
      [faux]
      While[p = RandomPrime[2^1024]; Mod[p - 1, 3] == 0];
      [pendant que | nombre premier aléatoire | modulo]
      While[q = RandomPrime[2^1024]; Mod[q - 1, 3] == 0];
      [pendant que | nombre premier aléatoire | modulo]
      n = p * q;
      dp = PowerMod[3, -1, p - 1];
      [puissance modulo]
      dq = PowerMod[3, -1, q - 1];
      [puissance modulo]
      m = RandomInteger[n - 1];
      [nombre entier aléatoire]
      s = ChineseRemainder[{PowerMod[m, dp, p], PowerMod[m, dq, q]}, {p, q}];
      [reste chinois | puissance modulo | puissance modulo]

In[10]= GCD[s^3 - m, n] == p
      [plus grand diviseur comm]

Out[10]= False
  
```

Figure 4.3: A False is displayed, as expected

```

RSAInvisible.nb * - Wolfram Mathematica 13.0
Fichier Édition Insérer Format Cellule Graphique Évaluation Palettes Fenêtre Aide

Flag = True;
      [vrai]
      While[p = RandomPrime[2^1024]; Mod[p - 1, 3] == 0];
      [pendant que | nombre premier aléatoire | modulo]
      While[q = RandomPrime[2^1024]; Mod[q - 1, 3] == 0];
      [pendant que | nombre premier aléatoire | modulo]
      n = p * q;
      dp = PowerMod[3, -1, p - 1];
      [puissance modulo]
      dq = PowerMod[3, -1, q - 1];
      [puissance modulo]
      m = RandomInteger[n - 1];
      [nombre entier aléatoire]
      s = ChineseRemainder[{PowerMod[m, dp, p], PowerMod[m, dq, q]}, {p, q}];
      [reste chinois | puissance modulo | puissance modulo]

In[10]= GCD[s^3 - m, n] == p
      [plus grand diviseur comm]
  
```

Figure 4.4: The initial notebook. Flag changed to True to activate the invisible formula.

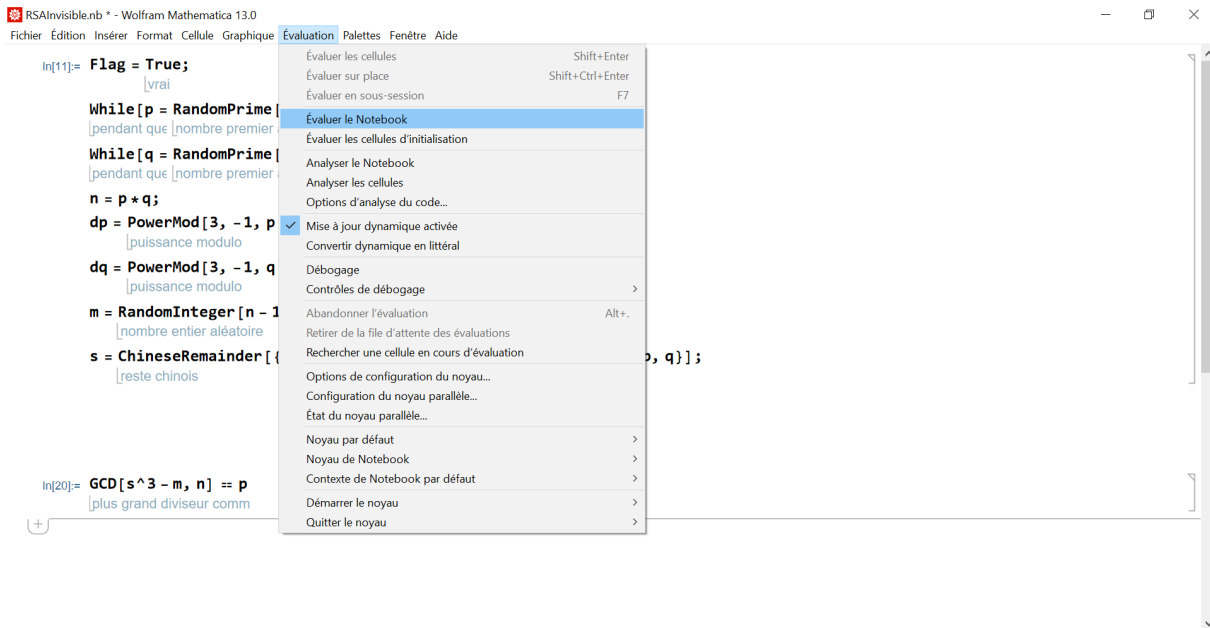


Figure 4.5: Executing the notebook again.

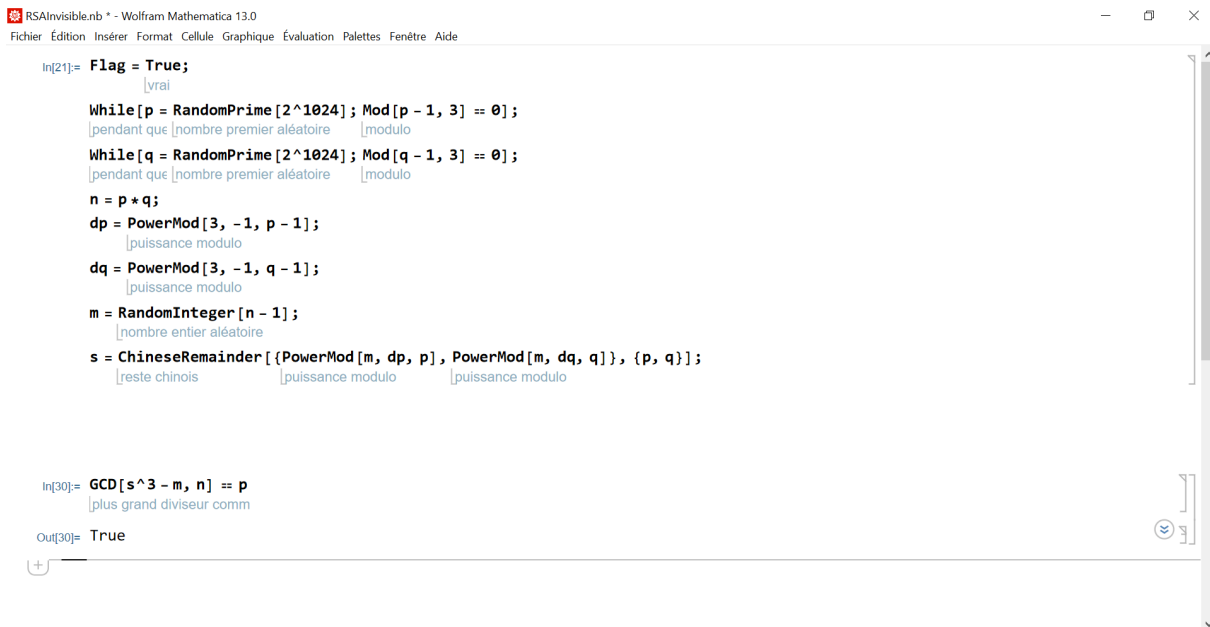


Figure 4.6: n is factored.

```

RowBox[{"While", "["],
  RowBox[{
    RowBox[{"p", "=",
      RowBox[{"RandomPrime", "["],
        RowBox[{"2", "^", "1024"}], "]"}}], ";",
    RowBox[{
      RowBox[{"Mod", "["],
        RowBox[{
          RowBox[{"p", "-", "1"}], ",", "3"}], "]"}, "==" , "0"}]]], "]"},
";"}], "\n",
RowBox[{
  RowBox[{"While", "["],
    RowBox[{
      RowBox[{"q", "=",
        RowBox[{"RandomPrime", "["],
          RowBox[{"2", "^", "1024"}], "]"}}], ";",
      RowBox[{
        RowBox[{"Mod", "["],
          RowBox[{
            RowBox[{"q", "-", "1"}], ",", "3"}], "]"}, "==" , "0"}]]], "]"},
";"}], "\n",
RowBox[{
  RowBox[{"n", "=",
    RowBox[{"p", "*", "q"}]}], ";"}], "\n",
RowBox[{
  RowBox[{"dp", "=",
    RowBox[{"PowerMod", "["],
      RowBox[{"3", ",",
        RowBox[{"-", "1"}], ",",
        RowBox[{"p", "-", "1"}]}], "]"}}], ";"}], "\n",
RowBox[{
  RowBox[{"dq", "=",
    RowBox[{"PowerMod", "["],
      RowBox[{"3", ",",
        RowBox[{"-", "1"}], ",",
        RowBox[{"q", "-", "1"}]}], "]"}}], ";"}], "\n",
RowBox[{
  RowBox[{"m", "=",
    RowBox[{"RandomInteger", "["],
      RowBox[{"n", "-", "1"}], "]"}}], ";"}], "\n",
RowBox[{

```

```

RowBox[{"s", "=",
  RowBox[{"ChineseRemainder", "["],
    RowBox[{"
      RowBox[{"{",
        RowBox[{"{
          RowBox[{"PowerMod", "["],
            RowBox[{"m", ",", "dp", ",", "p"}], "]"}, ",",
          RowBox[{"PowerMod", "["],
            RowBox[{"m", ",", "dq", ",", "q"}], "]"}}], "}"}, ",",
        RowBox[{"{",
          RowBox[{"p", ",", "q"}], "}"}}], "];"}], "Input",
(***** invisible code section start *****)
Cell[BoxData[
  RowBox[{"
    RowBox[{"If", "["],
      RowBox[{"Flag", ",",
        RowBox[{"s", "+=", "p"}], "]"}, "];"}], "Input",
ShowCellBracket->False,
ShowSelection->False,
CellBracketOptions->{"Color"->GrayLevel[1],
"HoverColor"->GrayLevel[0.1, 0.1],
"OverlapContent"->False},
PrivateCellOptions->{"ContentsOpacity"->0},
ShowCellLabel->False,
FontSize->2,
Magnification->0,
FontColor->GrayLevel[
  1]],
Cell[BoxData[""], "Text",
  ShowCellBracket->
  False],
(***** invisible code section end *****)
Cell[BoxData[
  RowBox[{"
    RowBox[{"GCD", "["],
      RowBox[{"
        RowBox[{"
          RowBox[{"s", "^", "3"}], "-", "m"}], ",", "n"}], "]"}, "=",
      "p"}], "Input"]
},
WindowSize->{582, 388},

```

```
WindowMargins->{{183.5, Automatic}, {Automatic, 39.5}}
]
```

The reader may object that the signatures produced by this code will not verify correctly and reveal the attack but it is very simple to evade such a detection using [Young, 1996]. If the PSS standard [Bellare, 1998] is used the invisible formula may encode a half of p 's bits in the salt to produce a perfectly standard signature from which the attacker can covertly extract p using [Coppersmith, 1996]. It is also possible to embed p in k unmodified signatures by re-generating signatures until the LSBs of each of those k signatures happen to encode a chunk of $\frac{\log_2 p}{2k}$ bits of p . For a 2048-bit n an invisible formula iterating the signature process $\simeq 256$ times per signature will leak p via 64 signatures.

1.4 Countermeasures

This note underlines the need to develop anti-malware tools adapted to mathematical software and/or provide easy-to-use interfaces restricting the operations performed by notebooks. For instance, Mathematica allows to run operating system commands², send emails³ or even connect to external services (e.g. Twitter, Facebook, Whatsapp etc) using the ServiceConnect command.

The problem is more acute when considering Paclet objects, Notebook Interfaces⁴ (that may spread invisible formulae to remote computers) or compiled Mathematica code, which is much harder to disassemble and analyze (more on this in a subsequent note). Because Mathematica is easier to use than C/C++, a number of developers write mathematical code in Mathematica and convert it automatically to C/C++ using the CCodeGenerator⁵. We witnessed this practice when custom or new algorithms (e.g. post-quantum) are concerned. Similarly, FortranForm is frequently used to automatically convert Mathematica to Python. If the Mathematica final code resorts to third party functions the risk of integrating invisible formulae must be taken into account. The same precaution applies to the use of Wolfram Symbolic Transfer Protocol (WSTP) to integrate Mathematica and C/C++ code.

An experiment allowing to assemble an executable Windows payload in a notebook upon execution and rootkit a target machine passed easily through 4 commercial email attachment scanners (as well as Gmail's standard scan). None of which blocked the concerned email. This payload encoder-decoder is purposely not published to avoid the scripting of real-world attacks.

Although a Mathematica notebook detecting the presence of invisible code (currently being developed by the authors) might reduce the attack surface, such empirical protections do not eliminate completely the threat. The tool, called WYSIWYX (standing for "What You See Is What You Execute") will rely on two detection techniques. The first is a symbolic analysis of the notebook aiming to detect invisible elements. The second opens the notebook with Mathematica, prints it into a PDF file, converts the PDF into a black and white bitmap, removes shot noise from the bitmap, OCR-converts the result to text and produces a new (hopefully safe) Mathematica notebook from the text.

We stress that our observation is not a vulnerability in Mathematica but rather a misuse of the rich possibilities offered by the software.

²such as Run, StartProcess, ProcessConnection, KillProcess.

³SendMail, SendMessage.

⁴<https://blog.wolfram.com/2021/12/13/new-in-13-notebook-interfaces/>

⁵<https://www.wolfram.com/mathematica/new-in-8/integrated-c-workflow/>

This calls for the formalization and the enforcement of security policies in such tools.

2 On Squaring Modulo Mersenne Numbers

Based on common work with David Naccache.

2.1 The Observation

During the design of a new lightweight primitive inspired by Squash [Shamir, 2008] we accidentally stumbled on the observation described in this short note.

The initial intent was to get an arbitrary input m of k bits whose entropy is $k' \leq k$ and hash m into a k -bit output c where each bit has entropy k'/k . A good candidate for doing so is modular squaring. In particular, working modulo a Mersenne number has notable computational advantages. This note shows that squaring modulo a Mersenne number does not provide this desirable entropy spreading property, even when $m^2 > n$ for some parameter configurations as some of the bits of c may depend *only* on specific bits of m .

The way in which this was accidentally discovered is interesting by its own right. The designed hash function was used as a building-block in an IoT malware analysis prototype. Packets including metadata and data were fed into a GAN that had to learn normal protocol behavior. Because part of the hashed data (the LSBs) consisted of constant system commands while the other part was a random nonce (the MSBs) to our surprise the GAN declared that part of the hash was... part of the protocol's semantics. This happened during the covariance detection phase where data is input into a filter reacting to the repeated appearance of sufficiently large patterns in the dataset. Looking into the reason for which this happened, we discovered the arithmetic phenomenon described in this note.

Let n be a k -bit Mersenne number $n = 2^k - 1$ whose factors are unknown. Consider an ℓ -bit secret number $x = 2^{k/2}a + b$. Although k is prime (and hence odd) we simplify it here as an even number to avoid managing unbalanced halves.

An attacker is given $c = x^2 \bmod n$. What can s/he learn about a and b individually?

We have:

$$c = x^2 = (2^{k/2}a + b)^2 = 2^k a^2 + 2^{1+k/2}ab + b^2 = 2^{1+k/2}ab + a^2 + b^2 \bmod n$$

Denoting $\Delta = ab$ and $\Gamma = a^2 + b^2$ we get

$$c = 2^{1+k/2}\Delta + \Gamma \bmod n$$

Γ is the modular sum of a k -bit number (b^2) and a $2\ell - k$ bit number (a^2). We observe that if $2\ell - k < k$, i.e. $\ell < k$, then Γ has good chances to be in \mathbb{Z} . Note that even if Γ exceeds n by one or two bits, those will wrap around and blur only a few LSBs of Γ leaving the remaining bits of $\Gamma \bmod n$ identical to those of Γ in \mathbb{Z} .

We now turn to analyzing the effect of adding to Γ the quantity $2^{1+k/2}\Delta$. We start by observing that Δ is of size ℓ . We distinguish in Δ two parts Δ_H (of size $k/2$) and Δ_L (of size $\ell - k/2$), i.e. $\Delta = 2^{\ell-k/2}\Delta_H + \Delta_L$.

We see that the addition of $2^{1+k/2}\Delta$ to Γ will blur the $\ell - k/2$ MSBs (because of Δ_L) and the $k/2$ LSBs (because of the wrapping of Δ_H). This will leave $k - \ell$ bits of Γ exposed.

Γ is essentially of size $2\log_2 b$ and is nothing but b^2 with its $2\ell - k$ (size of a^2) LSBs blurred. All in all it appears that c features the bits of b^2 between positions $\max(k/2, 2\ell - k)$ and $3k/2 - \ell$ which therefore depend only on b which, in our application, was a constant assortment of commands sent to the device.

It goes without saying that the home-made hash function was replaced by a standard Squash. This note confirms that the use of moduli of the form $2^k \pm r$ where r is small should always be analyzed carefully as episodically weaknesses due to this choice arise, e.g. [Borisov, 2002] or [Ouafi, 2009]⁶.

2.2 Example

Let $n = 2^{1009} - 1$ and fix randomly:

```

a = 00000004 b6b610e6 4e2d3680 139cca0b
b = 13fceaff 599d4f4e fa14b5c7 82d2f55c
    05c2c3ee 108fdd03 3f161099 237cb257
    24ac47a7 be03b21d d293d5e5 43e83374
    47dd3589 960fc891 669477c6 b7498278

```

Indeed, the quantities $c = (b + 2^{509}a)^2 \bmod n$ and b^2 coincide in their central bits as shown in red:

```

c = 1887fa50 303e3d1a c6c9b433 0e0087f4
    256fbc49 1d4628c7 7c45ca72 bbb65a96
    47c964b4 23ff555e 22cbea2f 5e8eaaca
    16eeabeb 7e988c3a cb3289e3 3136b061
    602e98ff dbd6560e e2d43566 aa9ef7b5
    6207638c 656dd780 5110d904 bfc4a799
    fe09cce3 01ba1cc 7bc61ac93 ec41c55b
    882cad79 cd602f49 ec00aa8f 3a06b
b2 = 184c165b d9601185 a8e14d91 ab8e0cfa
    0cac609f 8800030f 0327a865 e25c1d21
    957e2e15 cf5a290e 1fdaa07f bb68064c
    b5942217 ba885076 f8a3f8ba 440a1061
    602e98ff dbd6560e e2d43566 aa9ef7b5
    6207638c 656dd780 5110d904 bfc4a799
    fe09cce2 fef319ca a5a387bc 1473eb06
    7c6fd770 e258cdaf b8f433ae e1907

```

⁶Note that while very different, the observation in this note is somewhat reminiscent of [Ouafi, 2009] page 10, section 4.2.

3 On The Practical Advantage of Committing Challenges in Zero-Knowledge Protocols

Based on common work with David Naccache.

3.1 Introduction

Authentication is a cornerstone of information security, and much effort has been put in trying to design efficient authentication primitives.

The Fiat-Shamir transform is a classical technique for turning any zero-knowledge Σ -protocol into a signature scheme.

In essence, the idea underlying this transform is that deriving the challenge from the digest of the commitment suppresses simulatability and hence provides non-interactive proofs of interaction.

It follows from that observation that if one wishes to preserve deniability the challenge size (per round) must be kept low. For instance in the original Fiat-Shamir protocol the authors recommend 18 bits but suggest that the challenge size can be made larger to reduce communication overhead, e.g. the value of 20 is proposed in [Micali, 1990].

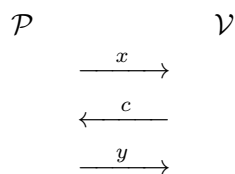
We show that even with relatively small challenge sizes *practical* deniability can be destroyed by having the verifier artificially impose upon himself the use of slowed-down hash function or by resorting to a trusted agency proposing an on-line deniability enforcement service against the provers community's will.

To that end, we will start by presenting generic notions and notations and later discuss our observation.

3.1.1 Σ -protocols

A Σ -protocol [Hazay, 2010; Damgård, 2010; Goldwasser, 1985] is a generic 3-step interactive protocol, whereby a prover \mathcal{P} communicates with a verifier \mathcal{V} . The goal of this interaction is for \mathcal{P} to convince \mathcal{V} that \mathcal{P} knows some value – without revealing anything beyond this assertion. The absence of information leakage is formalized by the existence of a simulator \mathcal{S} , whose output is indistinguishable from the recording (trace) of the interaction between \mathcal{P} and \mathcal{V} .

The three phases of a Σ protocol can be summarized by the following exchanges:



Namely,

- The prover sends a *commitment* x to the verifier;
- The verifier replies with a *challenge* c ;
- The prover gives a *response* y .

Upon completion, \mathcal{V} may accept or reject \mathcal{P} , depending on whether \mathcal{P} 's answer is satisfactory. Such a description encompasses well-known identification protocols such as Feige-Fiat-Shamir [Feige, 1988] and Girault-Poupard-Stern [Girault, 1990].

Formally, let R be some (polynomial-time) recognizable relation, then the set $L = \{v \text{ s.t. } \exists w, (v, w) \in R\}$ defines a *language*. Proving that $v \in L$ therefore amounts to proving knowledge of a witness w such that $(v, w) \in R$. A Σ -protocol satisfies the following three properties:

- *Completeness*: given an input v and a witness w such that $(v, w) \in R$, \mathcal{P} is always able to convince \mathcal{V} .
- *Special honest-verifier zero-knowledge*⁷: there exists a probabilistic polynomial-time simulator \mathcal{S} which, given v and a c , outputs triples (x, c, y) that have the same distribution as in a valid conversation between \mathcal{P} and \mathcal{V} .
- *Special soundness*: given two accepting conversations for the same input v , with different challenges but an identical commitment x , there exists a probabilistic polynomial-time extractor procedure \mathcal{E} that computes a witness w such that $(v, w) \in R$.

Many generalizations of zero-knowledge protocols have been discussed in the literature. One critical question for instance is to compose such protocols in parallel [Goldreich, 1991; Micali, 2006], or to use weaker indistinguishability notions (e.g., computational indistinguishability).

3.1.2 The Fiat-Shamir Transform

Hashing commitments is not a new idea: hashing x with a message m and using the result as c was used by Fiat and Shamir to purposely destroy deniability⁸. The Fiat-Shamir transform is a technique used to convert a zero-knowledge proof of knowledge (in particular Σ -protocols) into a digital signature scheme. The basic idea behind the transform is to replace the interaction between the prover and the verifier in the Σ -protocol with the use of a publicly computable function.

The function is constructed such that it takes as input the statement being proven, along with the challenge, and produces a response. The response, along with the statement, can then be used as a signature. The verifier can check the validity of the signature by re-computing the function using the statement and the response, and comparing the result to the original challenge.

In this way, the Fiat-Shamir transform allows one to convert a Σ -protocol, which only gives evidence of knowledge of a certain value, into a signature, which provides evidence of authenticity. The key benefit of this conversion is that the resulting signature scheme can be more efficient than a traditional zero-knowledge proof, since it eliminates the need for interaction between the prover and the verifier.

It is important to note that the Fiat-Shamir transform can only be applied to Σ -protocol where the proof is sound, that is if the verifier can efficiently check the correctness of the prover's proof.

3.2 Manufacturing Proofs of Interaction

The question around which revolves this section is that of *deniability*. A key feature of a ZKP is the fact that the verifier cannot bring a proof of interaction with the prover. This corollary of simulatability is useful in many

⁷Note that *special* honest-verifier zero-knowledge implies honest-verifier zero-knowledge.

⁸In a way the roots of this technique seem to even stretch back to ElGamal's celebrated signature scheme [ElGamal, 1985].

practical applications. Consider for instance a doctor’s card reader \mathcal{V} interacting with a patient’s contactless card allowing to get an anonymous methadone prescription. The card holder is entitled to get such services, however, for medical privacy reasons he does not want any proof to exist of such an interaction with the doctor \mathcal{V} and plausibly deny such interactions in case of need. A standard implementation of a ZKP perfectly answers this need: the doctor obtains the assurance that the patient is entitled for care whereas the patient leaves no traces after the consultation. Note that the card provided to the patient does not need to be anonymous, it can identify the patient (e.g. with a photo or a fingerprint) but leaves no traces.

Here we observe that for several parameter settings a malicious doctor \mathcal{V} may extract a proof of interaction from a card implementing a Σ -protocol.

The idea is that of purposely slowing down hashing. Consider a slowed-down version of a hash function H , denoted H^u . H^u consists in simply iterating u times the operation $H(x)$ to increase hashing time by a factor of u . We note that this does not contradict the theoretical *asymptotic* definition of zero-knowledge security, as it slows-down operations by a constant factor⁹. Yet it suffices to *practically* suppress deniability in several real-life parameter settings.

What happens if the verifier \mathcal{V} submits as a challenge $c = H^u(x)$ instead of a random c or $c = H(x)$?

A \mathcal{V} wishing to deprive \mathcal{P} from his deniability must exhibit a session trace x, c, y such that $c = H^u(x)$. Because c is of size k and each evaluation of H^u costs u it follows that the probability $P_k(w)$ that a \mathcal{V} investing a workload $w \times u$ can falsely pretend that \mathcal{P} participated in the session (while \mathcal{P} did not) is:

$$P_k(w) = 1 - \left(1 - \frac{1}{2^k}\right)^w$$

As a numerical application, let $u = 2^{40}$ and $k = 40$. A \mathcal{V} wishing to falsely pretend, with a success probability of 0.5, that \mathcal{P} participated in a session is expected to perform $w = 7.6 \times 10^{11} \times u \simeq 2^{39.5} \times u = 2^{79.5}$ hashing operations. This clearly puts the blame on \mathcal{P} .

With $u = 2^{45}$ and $k = 20$. A \mathcal{V} wishing to falsely pretend, with a success probability of 0.5, that \mathcal{P} participated in a session is expected to perform $w = 726817u \simeq 2^{19.5} \times u = 2^{64.5}$ hashing operations which, again, clearly puts the blame on \mathcal{P} .

Remark 34. The reader may object that there is an easy fix consisting in checking by \mathcal{P} that $H^u(x) \neq c$. This is unfortunately insufficient because \mathcal{V} can enrich hashing with a long secret random number r , unbeknownst to \mathcal{P} (in other words hash $c = H^u(\{x, r\})$) and later exhibit r as part of the proof.

Remark 35. Even the most succinct authentication protocols require collision-resistant commitments. Interestingly, while Girault and Stern [Girault, 1994] proved that breaking beyond the collision-resistance size barrier is impossible, a previous research [Ferradi, 2016] showed that if we add the assumption that the verifier can measure the prover’s response time, then commitment collision-resistance becomes unnecessary. The present work shows that measuring \mathcal{V} ’s response time can also be beneficial but for another goal: preserving the prover’s deniability.

3.2.1 Concurrent sessions

In a parallel ZKP the same \mathcal{P} sends ℓ commitments $x_0, \dots, x_{\ell-1}$ to one or several \mathcal{V} s and then gets ℓ challenges $c_0, \dots, c_{\ell-1}$ to which he answers with $y_0, \dots, y_{\ell-1}$.

⁹e.g. as is the case in [Merkle, 1987].

It is noted that if we derive the challenges $c_0, \dots, c_{\ell-1}$ by hashing $x_0, \dots, x_{\ell-1}$ then, again, deniability is broken if the entropy of $c_0, \dots, c_{\ell-1}$ is large enough (e.g. 80 bits). In this scenario we do not require a slowed-down H although slowing-down H can again serve to compensate for a smaller entropy in the $c_0, \dots, c_{\ell-1}$. This works even if each c_i is a single bit.

A large number of papers was published on concurrent zero-knowledge, we recommend to the reader the excellent state of the art reference [Pass, 2015].

3.2.2 Using a trusted deniability enforcement agency

In this section we assume that a national agency \mathcal{A} opposed to undeniability proposes an online service allowing verifiers to obtain and keep proofs of interaction with provers. We note that such a service can also be proposed by an association opposed to undeniability for ideological reasons. Here \mathcal{A} is not opposed to the fact that provers are able to prove their identities, but proposes a service allowing verifiers to generate proofs of their interactions with the provers even without the provers' consent or knowledge. It is important to underline that \mathcal{A} does not “cheat” or “collude” in any way but just honestly performs the service it advertises to the verifiers' community.

When \mathcal{V} gets a commitment x from a prover \mathcal{P} , \mathcal{V} forwards x and \mathcal{P} 's public-key (denoted $\text{pk}_{\mathcal{P}}$) to \mathcal{A} .

\mathcal{A} keeps a table counting the number of requests performed by verifiers for each $\text{pk}_{\mathcal{P}}$, we denote this counter by $\omega_{\mathcal{P}}$. When an $\omega_{\mathcal{P}}$ exceeds a limit η , \mathcal{A} will stop answering queries concerning $\text{pk}_{\mathcal{P}}$. The goal of the bound η is to prevent verifiers from forging proofs of interaction without actually interacting with targeted provers.

If $\omega_{\mathcal{V}} < \eta$, \mathcal{A} will increase $\omega_{\mathcal{P}}$ and answer the query with a signature σ on the data $x, \text{pk}_{\mathcal{P}}$.

\mathcal{V} will keep σ as a proof and derive the challenge c from σ by hashing.

At a later stage, \mathcal{V} can exhibit σ and prove the interaction under the hypothesis that \mathcal{A} played by the rules. Indeed \mathcal{A} will not sign more than η commitments per verifier and this rules-out the possibility of exhaustive search by \mathcal{V} .

Evidently, if a central \mathcal{A} is insufficient in terms of public trust, \mathcal{A} can be replaced by any group signature involving a multiparty protocol that ascertains that several agencies or entities collaborated to produce σ . The odds that *all* such agencies cheat diminishes the probating value of \mathcal{P} 's future deniability claims.

For $k = 40$ and $\eta = 2^{20}$ (i.e. the possibility to use \mathcal{A} 's services one million times per prover), the odds that a \mathcal{V} exhibiting a proof of interaction is falsely accusing \mathcal{P} drops to 2^{-20} . Thereby, again, destroying \mathcal{P} 's undeniability without even \mathcal{P} knowing about this.

Remark 36. To avoid resistance movements from flooding \mathcal{A} with signature requests associated to a given $\text{pk}_{\mathcal{P}}$ and hence protect \mathcal{P} ¹⁰, \mathcal{A} may charge a fee for each signature and/or request \mathcal{V} to identify himself and blacklist dishonest \mathcal{V} 's as soon as those provides too many incorrect y_i 's corresponding to the x_i 's on which they requested \mathcal{A} 's signatures. This opens yet another resistance strategy on \mathcal{P} 's behalf consisting in purposely failing authentication attempts. A fix can be implemented by having verifiers refuse to identify any \mathcal{P} whose $\omega_{\mathcal{P}}$ reached η (i.e. a \mathcal{P} for which \mathcal{A} denies signatures). Hence we see that this scenario is hybrid adversarial scenario involving both information security measures and cryptographic strategies. Yet in real-world settings it can be problematic and of practical significance.

¹⁰by pushing artificially the counter $\omega_{\mathcal{P}}$ to the limit η .

3.3 Mitigation

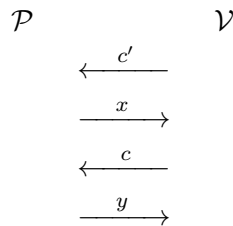
Protections against those deniability deprivation scenarios revolve around three mitigation measures:

- Reduce k (preferably to say less than 8 bits) so as to preserve simulatability while preserving \mathcal{P} 's efficiency.
- Add to the protocol the extra specification that a time-out between the sending of x and the receiving of c must be respected. If such a time-out is not respected \mathcal{P} is instructed to abort and not send y .
- Concurrent sessions should be avoided, i.e. \mathcal{P} should agree to open a new session only when the previous is over (or consider a current session as interrupted¹¹ as soon as a new challenge y_{i+1} arrives).

In particular, countering the trusted deniability enforcement agency scenario requires reducing k and repeating the protocol to achieve the desired security level.

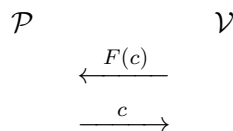
3.4 A generic mitigation

A further, less trivial and more generic mitigation, consists in banning concurrent interactions and modifying the protocol as follows:



Here c' is a commitment on c . This prevents \mathcal{V} from feeding \mathcal{P} with a doctored c and allows using any k because \mathcal{P} will abort¹² if c does not correspond to the c' received at the protocol's start. If the entropy of c is low \mathcal{V} can generate a sufficiently long random r and define $c' = H(r, c)$ with r being revealed at the commitment opening stage (this requires r to be added along with c and the third exchange).

This idea can also be used to derive a deniable zero-knowledge mode of operation for any public-key cryptosystem (denoted F and F^{-1}). Consider first the following protocol:



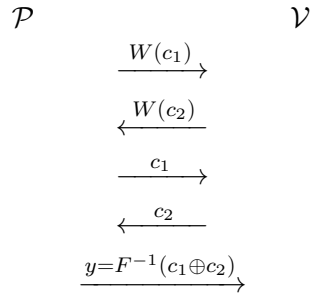
This protocol is obviously not zero-knowledge because \mathcal{V} may select as $F(c)$ a number presenting a redundancy which destroys deniability. A first solution consists in selecting c featuring a redundancy and have \mathcal{P} check that c was indeed chosen honestly before returning it. This setting is trivial to simulate but it requires a random oracle or specific assumptions on the padding function used to introduce redundancy into c .

A more elegant approach not resorting to random oracles consists in jointly agreeing about a common challenge $c_1 \oplus c_2$. There are two ways of doing so. We will now use two one-way permutations U, W that can

¹¹i.e. \mathcal{P} will not agree to send the y_i anymore.

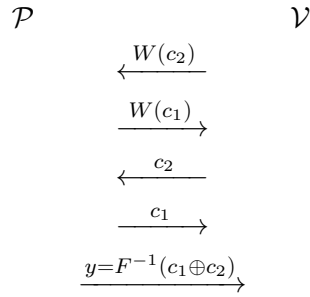
¹²i.e. not send y .

be of the same family as F to avoid requiring additional complexity assumptions¹³. The first (insecure) protocol is:



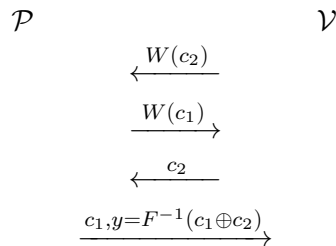
It is easy to see that this does not guarantee deniability, indeed, having received \mathcal{P} 's commitment first \mathcal{V} may manufacture $c_2 = U(W(c_1))$, which would result in a $y = F^{-1}(c_1 \oplus U(W(c_1)))$. Because \mathcal{V} is unable to invert F his only solution is to pick a random pre-image α and solve the equation $\alpha = x \oplus U(W(x))$ which is impossible. Hence exhibiting a solution demonstrates that using \mathcal{P} is the only way in which the proof of interaction was obtained, which in turn destroys \mathcal{P} 's deniability.

Consider now the same protocol in which \mathcal{V} speaks first:



The situation is now radically different. We see that, having committed himself on c_2 ¹⁴ \mathcal{V} must work with c_2 into which he cannot inject any information coming from \mathcal{P} in subsequent deniability destruction attempts.

We can now note that the last steps of the above protocol are communications from \mathcal{P} to \mathcal{V} and hence simplify the protocol into a 4-move one:



To simulate the protocol, \mathcal{V} can pick in advance a random \bar{y} , compute $F(\bar{y})$, pick a random \bar{c}_1 and compute $c_2 = \bar{y} \oplus \bar{c}_1$. \mathcal{V} is now able to complete the simulation by computing the commitments $W(\bar{c}_1)$, $W(c_2)$ to output:

¹³There remains the technical question of generating the public parameters for U, W that we skip here as there are several algorithmic ways to do so. For instance in the case of RSA very long moduli extracted from a public constant such as π can be used thereby ascertaining that with high probability roots cannot be computed by anybody.

¹⁴and given that the protocol will fail if this commitment is subsequently found to be false by \mathcal{P}

$$\{W(\bar{y} \oplus \bar{c}_1), W(\bar{c}_1), \bar{y} \oplus \bar{c}_1, \bar{y}\}$$

We now note, as a last simplification step, that the transmission of c_1 at the last step is superfluous. Indeed, \mathcal{V} can easily derive from y the quantity $c_1 \oplus c_2$ and, knowing c_2 , derive c_1 . He can hence check $W(c_1)$ and complete the protocol. This results in the simplified version:

$$\begin{array}{ccc}
 \mathcal{P} & & \mathcal{V} \\
 & \xleftarrow{W(c_2)} & \\
 & \xrightarrow{W(c_1)} & \\
 & \xleftarrow{c_2} & \\
 & \xrightarrow{y=F^{-1}(c_1 \oplus c_2)} &
 \end{array}$$

3.5 Conclusion

In this section we underlined the practical risk that stems from the use of too long challenges in zero-knowledge protocols. We show that for practical purposes, even 20 or 40 bit challenges can result in situations where the prover's deniability is compromised. A generic solution, ascertaining that the challenge was chosen randomly seems to cleanly settle the issue and, given its simplicity, we recommend to implement it in practical settings where deniability is of importance.

4 Authenticating Medications with QR-Codes and Compact Digital Signatures

Based on common work with Julien Jainsky, Bassem Ouni and David Naccache.

4.1 Introduction

A recent article¹⁵ [Overstreet, 2019] reports that a the \$200 billion pharma counterfeit drug market is growing by 20% *per annum*.

The issue of fake medications poses a significant and widespread global concern, endangering the health and well-being of countless individuals. According to the World Health Organization (WHO) [World Health Organization, 2017], approximately 10.5% of medicines available worldwide may be counterfeit with this figure reaching an alarming levels in some regions. For example, In 2017, the WHO reported issues with 33.6% of hypertension, cancer, epilepsy, analgesic uterotronics and immunosuppressants drugs from 75 low- and middle-income countries (LMIC) [World Health Organization, 2017]. On top of these, it is estimated that approximately 50% of the drugs sold via the internet are fake [Clark, 2015]. These counterfeit drugs not only fail to provide the intended therapeutic benefits but can also lead to adverse health effects, drug resistance, and even fatalities.

These revelations serve as a resounding call to action, emphasizing the imperative need for robust product verification and tracking capabilities within the healthcare realm. Hence, any cheap technological solution allowing to control or mitigate the problem is welcome.

4.2 The solution

We seek to design a blister packaging solution which is cheap to manufacture, easy to check electronically and allows patients and pharmacists to instantly detect fakes. Ideally, such a solution should not include a chip in the medication's package (as this is costly) and rely on an application running on the patient's mobile phone.

Under such constraints, what comes to mind naturally is the use of QR codes, digital signatures and some unique hardly reproducible physical features. We will overview the different components of the proposed solution and combine them to reach the desired goal.

4.2.1 Drawing inherent randomness

Using the inherent characteristics of disordered systems is not new at all and solutions leveraging this idea were re-invented over and over again. In 1983, Bauder [Bauder, 1983] made one of the earliest documented references to such systems, followed closely by Simmons in 1984 [Simmons, 1984; Simmons, 1991]. Building on these pioneering works, Naccache and Frémanteau introduced an authentication scheme specifically tailored for memory cards [Naccache, 1992]. We hence naturally looked for already existing inherent randomness in the packaging process. We will describe here two such ideas.

4.2.1.1 Two colored pills.

Current packaging techniques such as the one shown in Figure 4.7 provide some randomness. However, given

¹⁵<https://bit.ly/3BZPWPE>



Figure 4.7: A 10-bit random pattern formed naturally during packaging.

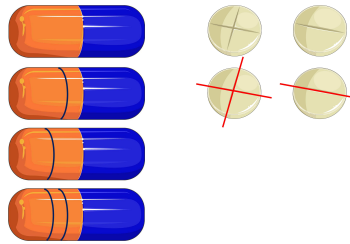


Figure 4.8: Using $k = 2$ stripes to encode information in capsules (left) and Diameter detection (right). Source: https://smart.servier.com/smart_image, modified by the authors.

that pills are usually packed by $n = 10$ to 20 relying on the pills' orientation alone does not provide enough entropy: Let T be the number of genuine packages needed to collect all 2^n pill combinations. It is known (coupon collector's problem) that:

$$E(T) = 2^n n \log 2 + \gamma 2^n + \frac{1}{2} + O(2^{-n}) \text{ and } \Pr(|T - E(T)| \geq c 2^n) \leq \frac{\pi^2}{6c^2}$$

where $\gamma \simeq 0.5772$ is the Euler–Mascheroni constant.

Entropy can be cheaply increased (Figure 4.8, left), by randomly decorating each pill with k bars on one of the pill's sides. This adds 2^{k+1} bits per pill and an overall entropy of $2^{(k+1)n-1}$ bits per package¹⁶. For $(k, n) = (3, 10)$ we get 2^{39} combinations and an $E(T) \simeq 2^{39} \log 2^{39} \simeq 2^{43.76}$.

This solution offers only a modest form of security as a moderately sophisticated fraudster could still come-up with a manufacturing process placing the right pills in the right order to match a configuration copied from a genuine package.

4.2.1.2 Orientation in circular pills.

Another simple method consists in using the diameter naturally present in most pills as an angle encoding information. If this method is chosen, the packaging should be tight enough to forbid pills from spinning around after packaging. This is illustrated in Figures 4.8 (right) and Figure 4.9.

The detection of the pills' orientation is easy to extract using existing image processing tools. In our experiment, we placed 8 Prednisone pills on a black surface and photographed them using a common Samsung A5 smartphone.

¹⁶The -1 in the exponent comes from the fact that by rotating a package upside-down one more combination can be gained by the forger.



Figure 4.9: 30 pills encoding information using diameter orientation (illustration). Source: https://smart.servier.com/smart_image, modified by the authors.

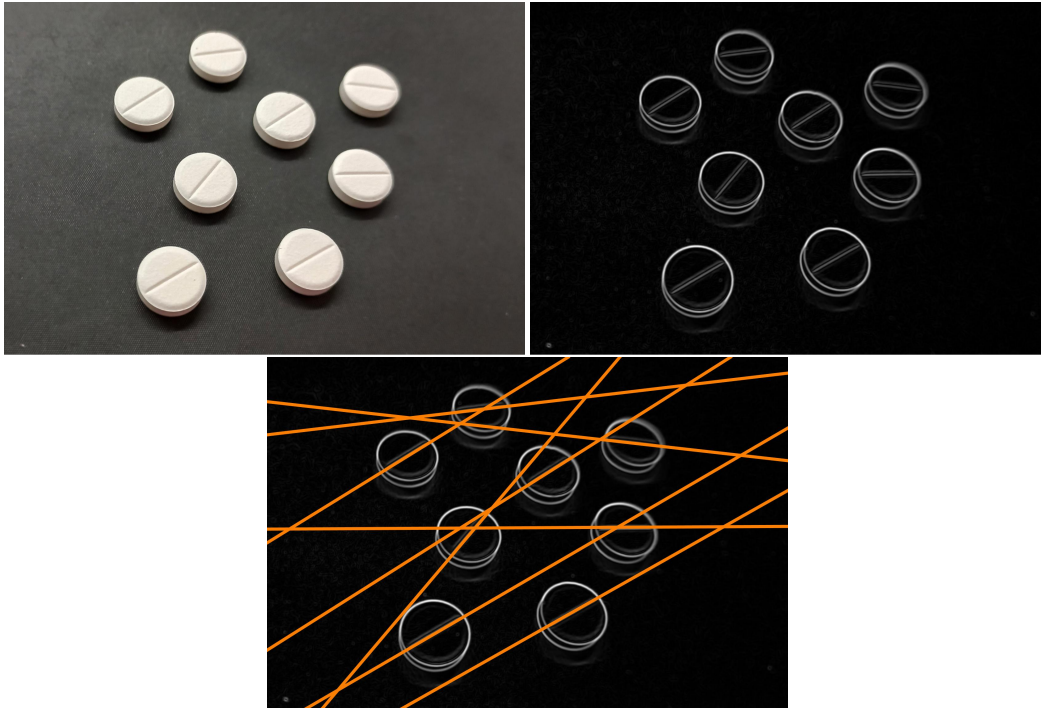


Figure 4.10: Pill identification attempt in the absence of artefacts.

The resulting image was named `image0.png`. `image0` was passed through a gradient filter¹⁷ to generate `image1`. We then extracted 8 lines from `image1`¹⁸ and superimposed the extracted lines on `image1` to get `image2`¹⁹. Indeed, all the angles were easily detected. Repeating the experiment (with `MaxFeatures->18`) in the presence of artefacts proved insufficient and required further filtering but such artefacts will not exist during field deployment.

The industrialization of this solution requires some easy technical refinements to deal with borderline angles using error correction on the signed data embedded into the QR-code and seems much harder to circumvent.

4.2.2 Packaging and QR-code printing

A QR can be either printed on the back of the blister package or on the back of the paper box containing the medications if the box is equipped with a transparent plastic window (such as the one shown in Figure 4.12) allowing the scanning of the QR code from outside the box using the smartphone.

If a standard box is used, we recommend to use micro QR-codes that can store up to 128 bits of information, such codes are shown in Figure 4.13. Such a solution requires compressing the signature on the inherent randomness into 16 bytes or spreading the signature over several micro QR-codes.

Ideally, a second (constant) QR-code present on the box would allow the patients to install the application, thereby avoiding version issues.

An option, that we do not recommend, is to encode in the QR-code a URL redirecting to a digital signature stored online. Note that an online digital signature database is not expected to grow indefinitely as it could be

¹⁷`image1=GradientFilter[image0,10]//ImageAdjust`

¹⁸`lines=ImageLines[EdgeDetect[image1],MaxFeatures->8]`

¹⁹`image2=HighlightImage[image1,Orange,lines]`

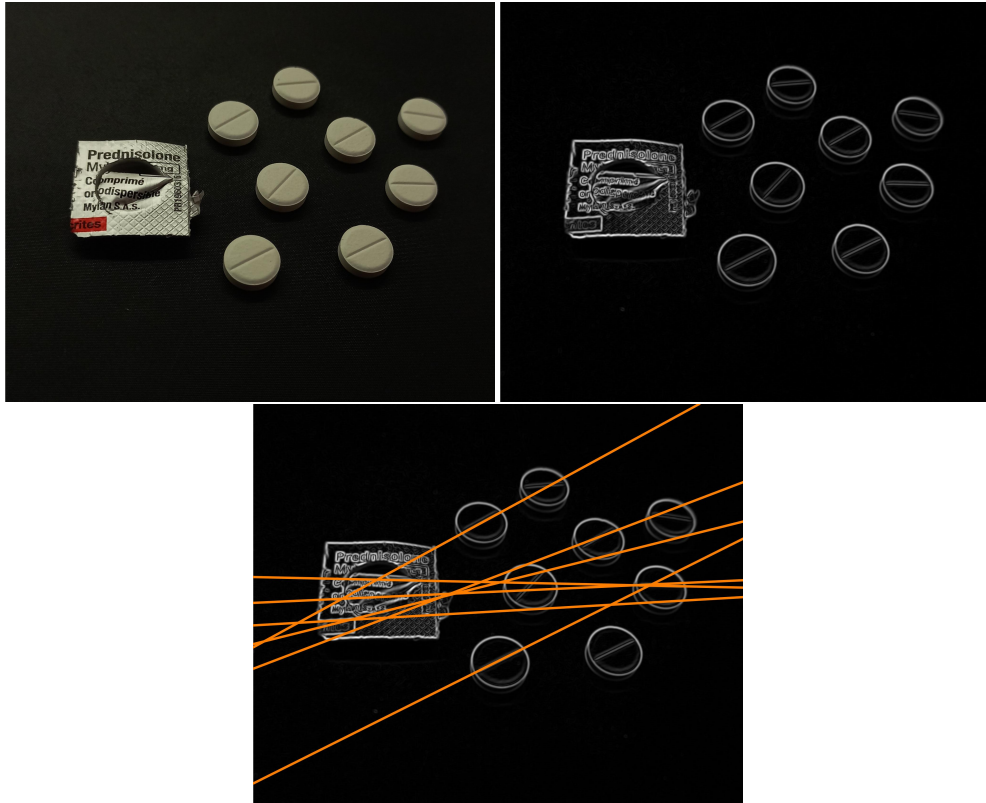


Figure 4.11: Pill identification attempt in the presence of artefacts.



Figure 4.12: Simple paper box with a window.



Figure 4.13: Micro QR-codes on medications printed on a medication package. Source: <http://www.chinatimes.com/news/papers/20140805000894-260113>

sanitized when medications expire. This solution has the additional advantage of allowing to count the number of accesses to any given signature and hence blacklist copied URLs after too many verifications (e.g. 10). We discard this solution as it requires an online communication which might not always be available.

4.3 Short signatures

The current record in terms of signature size seems to be 110 bits, held by [Mohamed, 2016]. Truncating signatures to reduce their size was treated previously by [Naccache, 2001] and [Pornin, 2022], resulting in shorter DSA-like signatures without loss of security. Using those approaches, signature size is linearly reduce at the cost of additional exponential computations on the signer and/or the verifier side.

[Naccache, 2001] proposed a solution for reducing the size of the DSA-like signatures by 2^ℓ bits at an $O(2^\ell)$ work by signer and by the verifier. Typically $32 \leq \ell \leq 40$ bits. [Pornin, 2022] improves this by requiring the 2^ℓ effort to be done only at the verifier’s side. As [Pornin, 2022] “frees” the signer again, we can now have the signer make a 2^ℓ effort to squeeze ℓ more bits by varying the DSA nonce k and searching for a short r . Note that because r does not depend on the message, a library of “good” r values could be constructed offline and used upon signing. Regularizing the flow of such r values during production can be important and there are known techniques for doing so, e.g. [Ferradi, 2015].

All in all, we can hence achieve a 3ℓ shortening gain at the cost of $O(2^\ell)$ operations by the signer and the verifier.

A typical EC-DSA signature is 56 bytes long, which means that choosing $\ell = 40$ yields a 41 byte signature. Legacy DSA produces 40 bytes signatures, in which case, with $\ell = 40$ bits will shorten the signature size to 25 bytes.

Note that another interesting way of shortening DSA-like signatures (to the best of our knowledge not reported so far) is the following: The signer generates 2^ℓ elements r and stops when a specific r is found. The form of this r is the following something $|\alpha|$ where α is any ℓ -bit string. Because there are 2^ℓ possible α values a good r is expected to be found in $O(2^\ell)$. By transmitting only the “something” part, the verifier can, using 2^ℓ verifications, retrieve α and verify the signature. This alternative to the discrete logarithm approach of [Pornin, 2022] shortens a signature by 2ℓ bits at the cost of $O(2^\ell)$ work by both parties and its constant factor might prove smaller than the constant factor of [Pornin, 2022] (unchecked). In addition DSA verifications lend themselves to batching which might also result in some constant gains [MRaihi, 1996].

4.4 Conclusion & an open question

We have described a way to protect medications against falsification, a long-standing problem in the world. The proposed solution does not require the inclusion of chips in packages and relies on cheap existing technologies. The building-blocks used are inherent physical randomness generated during the packaging process, artificial vision, short digital signatures and QR-codes.

From a conceptual standpoint, the following question remains: *Given the collection of signature shortening ideas published so far can a Schnorr-like signature be shortened by more than 3ℓ bits at the cost of $O(2^\ell)$ effort per party without loss of security?*

We conjecture that such is not the case given that all our attempts to combine different 2ℓ solutions ended-up in a total gain of 3ℓ at best. [Neven, 2009] is a useful reference to consult in that respect.

5 Fiat-Shamir Goes Tropical

Based on common work with Rémi Géraud-Stewart and David Naccache.

5.1 Introduction

This section can be read as a continuation of a thread of papers on a tropical signature scheme proposed by Chen, Grigoriev and Shpilrain (CGS) [Chen, 2023]. We refer the readers to [Brown, 2023; Kim, 2005; Panny, 2023], for the previous episodes of this saga. Further useful references on the topic are given in [Muanalifah, 2021] and its bibliography.

5.1.1 Chen–Grigoriev–Shpilrain signature

Denote by $S[t]$ the set of polynomials in t , with $(\min, +)$ as addition and multiplication respectively, which we henceforth write \oplus and \otimes respectively. $S[t]$, extended with a multiplicatively neutral element ϵ , is called the tropical polynomial semiring. Denote by $\mathcal{P}_{r,d}$ the subset of tropical polynomials with coefficients in $[0, r]$ and degree d .

Key generation: The signer chooses r, d appropriately and selects $X, Y \in_R \mathcal{P}_{r,d}$. The public key is:

$$\text{pk} := (r, d, M = X \otimes Y).$$

Signature: The signer hashes a message m into $H \in \mathcal{P}_{r,d}$, picks $U, V \in_R \mathcal{P}_{r,d}$, and computes the signature:

$$\begin{aligned} \sigma &:= (H, H \otimes X \otimes U, H \otimes Y \otimes V, U \otimes V) \\ &= (H, A, B, N). \end{aligned}$$

Verification: Given $\sigma = (H, A, B, N)$ and m , the following verifications are performed:

- V1: $\text{hash}(m) \stackrel{?}{=} H \in \mathcal{P}_{r,d}$
- V2: $A, B \stackrel{?}{\in} \mathcal{P}_{3r,3d}$
- V3: $N \stackrel{?}{\in} \mathcal{P}_{2r,2d}$
- V4: Neither A nor B is a constant tropical multiple of $H \otimes M$ or $H \otimes N$.
- V5: $A \otimes B \stackrel{?}{=} H \otimes H \otimes M \otimes N$

If any step fails then the signature is considered invalid, otherwise it is considered valid.

5.1.2 Security and fix

As pointed out in the references above, the Chen–Grigoriev–Shpilrain (CGS) signature is insecure, as there are multiple attacks allowing for forgery. Our approach consists in using a tropical version of the Fiat–Shamir identification scheme as a building block to re-engineer CGS and dodge all known attacks. In doing so, we hope to redirect attacks on the key and on tropical polynomial factorization, which is believed to be hard in general.

5.2 Tropicalized Fiat-Shamir

The first fix is interesting in that it translates directly a classical factoring-based scheme into a similar (hopefully) post-quantum scheme.

5.2.1 Standard Fiat-Shamir protocol

We use here the standard notations of [Fiat, 1987]. The problem with the classical Fiat-Shamir is that obtaining $s^2v = 1 \pmod n$ requires a modular inversion during key generation. It is easy to work around this limitation by defining $s^2 = v \pmod n$ instead, which is equivalent up to relabeling s by its inverse, and results in the following scheme:

1. The prover starts by picking randomly an $r \in_R \mathbb{Z}$;
2. The prover sends a commitment $x = r^2 \pmod n$;
3. The verifier replies with a challenge bit b ;
4. The prover responds with $y = s^b r \pmod n$;
5. The verifier checks that $y^2 \stackrel{?}{=} v^b x \pmod n$.

5.2.2 Tropical Fiat-Shamir protocol

We can now translate directly: the secret key becomes $S \in_R \mathcal{P}_{r,d}$, and the public key becomes $V = S \otimes S \in \mathcal{P}_{2r,2d}$. We introduce an auxiliary selection function for $L_0, L_1 \in \mathcal{P}_{*,*}$ and $b \in \{0, 1\}$: $\Delta_b(L_0, L_1) = L_b$. Here's the protocol:

1. The prover starts by picking randomly an $R \in_R \mathcal{P}_{r,d}$;
2. The prover sends a commitment $X = R \otimes R$;
3. The verifier replies with a challenge bit b ;
4. The prover responds with $Y = \Delta_b(R, S \otimes R)$;
5. The verifier checks if $Y \otimes Y \stackrel{?}{=} \Delta_b(X, V \otimes X)$.

The verifier also checks that $V, X \stackrel{?}{\in} \mathcal{P}_{2r,2d}$ and $Y \stackrel{?}{\in} \mathcal{P}_{(1+b)r,(1+b)d}$.

5.2.3 Signature from TFS

To get a signature scheme from this zero-knowledge protocol one can just apply the Fiat-Shamir transform.

We note that the Chen, Grigoriev and Shpilrain differs from the above protocol in two points: the first is that it corresponds to a tropicalized Fiat-Shamir where the challenge b is always stuck to 1. The second is that squares are not used but the operation r^2 is replaced by $r_1 r_2$.

Table 4.1: Protocol modification to accommodate simulation.

when the challenge is $b = 1$	S	V	R	X	Y
legitimate protocol	$\mathcal{P}_{r,d}$	$\mathcal{P}_{2r,2d}$	$\mathcal{P}_{2r,2d}$	$\mathcal{P}_{4r,4d}$	$\mathcal{P}_{3r,3d}$
simulator	$\mathcal{P}_{r,d}$	$\mathcal{P}_{2r,2d}$	$\mathcal{P}_{r,d}$	$\mathcal{P}_{4r,4d}$	$\mathcal{P}_{3r,3d}$

Table 4.2: Protocol modification to accommodate simulation.

when the challenge is $b = 0$	S	V	R	X	Y
legitimate protocol	$\mathcal{P}_{r,d}$	$\mathcal{P}_{2r,2d}$	$\mathcal{P}_{2r,2d}$	$\mathcal{P}_{4r,4d}$	$\mathcal{P}_{2r,2d}$
simulator	$\mathcal{P}_{r,d}$	$\mathcal{P}_{2r,2d}$	$\mathcal{P}_{2r,2d}$	$\mathcal{P}_{4r,4d}$	$\mathcal{P}_{2r,2d}$

5.2.4 Security

We do not know how to simulate the tropicalized Fiat-Shamir for the following reason. Following the traditional *modus operandi* the case $b = 0$ is trivial. For $b = 1$ we would use $X = R \otimes R \otimes V$ and $Y = R \otimes V$:

$$Y \otimes Y \stackrel{?}{=} V \otimes X$$

Indeed:

$$(R \otimes V) \otimes (R \otimes V) = V \otimes (R \otimes R \otimes V)$$

However, now $X \in \mathcal{P}_{4r,4d}$ and $Y \in \mathcal{P}_{3r,3d}$ which violates the verification conditions. This does not mean that the tropicalized Fiat-Shamir version is insecure but only that, do date, we don't know how to simulate it and prove that it is zero-knowledge.

A potential way to get around this problem might be to increase in the legitimate protocol specifications to $R \in \mathcal{P}_{2r,2d}$. In which case the simulator could use “shorter than normal” R s and the situation will be:

While this fixes the size problem, nothing guarantees that the (X, Y) distributions of the parties and of the simulator are strictly identical. Nonetheless this gives hope to prove the protocol secure in the statistical zero-knowledge rather than in the perfect zero-knowledge sense. We did not explore further this point.

Note that while CGS security relied on the conjectured hardness of factoring polynomials in $S[t]$, the Fiat-Shamir variant relies on both the hardness of factoring polynomials in $S[t]$ and on the conjectured hardness of computing square roots in $S[t]$, a problem which might turn out to be easier than factoring polynomials in $S[t]$ ²⁰.

5.3 Fiat-Shamirization of Chen, Grigoriev and Shpilrain

The fixing strategy will consist in translating the Fiat-Shamir protocol into the tropical realm (first fix) and by applying the Fiat-Shamir Transform (FST) to CGS (second fix).

For the sake of convenience we will denote for any collection of objects L_i :

$$\vec{L} = (L_0, L_1, \dots, L_{\tau-1})$$

Any operation \star between arrowed variables is to be understood component wise, e.g.

²⁰Note, for instance that $2P(0) = (P \otimes P)(0)$ and that the same occurs on the leading coefficients of both P and $P \otimes P$. It is unclear if this can be used to unravel P from $P \otimes P$.

$$\vec{L} \star \vec{L}' = (L_0 \star L'_0, L_1 \star L'_1, \dots, L_{\tau-1} \star L'_{\tau-1})$$

We denote $\text{hash}(m, i) = H_i \in \mathcal{P}_{r,d}$.

Key generation remains unchanged with respect to the original CGS.

To sign a message, generate $\vec{U}, \vec{V} \in_R (\mathcal{P}_{r,d})^\tau$ and compute $(\vec{A}, \vec{B}, \vec{N})$ as in the original CGS. Let $\vec{C} = \vec{A} \otimes \vec{B}$. Here \vec{C} will act as a non-interactive commitment on \vec{A} and \vec{B} .

$$h = \text{hash}(r, d, m, M, \vec{C}, \vec{N}) \bmod 2^\tau$$

We start by including in the signature \vec{N} .

We now use the τ bits of h as indicators pointing which commitments to open.

$$\begin{cases} \text{if } h_i = 0 & \text{add to the signature } U_i, V_i, C_i \\ \text{if } h_i = 1 & \text{add to the signature } A_i, B_i \end{cases}$$

The verifier can hence reconstruct \vec{C} in full. Either they get A_i, B_i and can hence compute C_i (case $h_i = 1$) or they get C_i directly (case $h_i = 0$). The verifier can hence recompute h from the signature.

At each $h_i = 1$ coordinate the verifier performs tests V_1, V_2, V_3, V_4 and V_5 .

At each $h_i = 0$ coordinate the verifier checks that $U_i, V_i \in \mathcal{P}_{r,d}$ and $N_i \in \mathcal{P}_{2r,2d}$.

This idea can come in several flavors e.g. a different X_i, Y_i can be used per coordinate. In a more daring variant we can aggregate the different signature components.

In this variant we modify the definition of h to:

$$h = \text{hash}(r, d, m, M, \vec{C}, \vec{N}) \bmod 2^\tau \text{ where } \vec{C} = \bigotimes_{i=0}^{\tau} C_i \text{ and } \vec{N} = \bigotimes_{i=0}^{\tau} N_i$$

Let:

$$\vec{A}^1 = \bigotimes_{h_i=1} A_i \text{ and } \vec{B}^1 = \bigotimes_{h_i=1} B_i \text{ and } \vec{C}^0 = \bigotimes_{h_i=0} (A_i \otimes B_i) = \bigotimes_{h_i=0} C_i$$

$$\vec{N}^0 = \bigotimes_{h_i=0} N_i \text{ and } \vec{N}^1 = \bigotimes_{h_i=1} N_i \text{ and } M^\tau = \bigotimes_{i=0}^{\tau-1} M$$

If the signature was correctly generated we should have:

$$\vec{A}^1 \otimes \vec{B}^1 \otimes \vec{C}^0 = \vec{N} \otimes M^\tau \otimes \bigotimes_{i=0}^{\tau} (H_i \otimes H_i)$$

$$\vec{A}^1 \otimes \vec{B}^1 \otimes \vec{C}^0 = \vec{N}^0 \otimes \vec{N}^1 \otimes M^\tau \otimes \bigotimes_{i=0}^{\tau} (H_i \otimes H_i)$$

We can hence provide as a signature:

$$\vec{A}^1, \vec{B}^1, \vec{C}^0, \vec{N}^1, \text{ plus all the } U_i, V_i \text{ couples for which } h_i = 0$$

How to generate U_i, V_i ? In all the above we assumed for the sake of clarity that \vec{U}, \vec{V} are randomly generated.

The second (aggregated) variant uses \otimes as a hash function which is bad because \otimes is commutative. We hence enforce an extra protection to thwart element permutation attacks (see e.g. [Benhamouda, 2017]). The protection consists in generating each U_i, V_i pair from a common random seed σ_i and by including in the process transforming σ_i into U_i, V_i both σ_i **and the index i** . To reveal a given U_i, V_i pair the signer reveals σ_i . This protection is mandatory in the aggregated scheme and recommended as an extra precaution for the non-aggregated version.

5.4 Tropicalizing other cryptosystems

The “tropicalization” strategy described for Fiat-Shamir case *may* apply to a variety of classical cryptosystems as long as during all computations²¹ the following holds:

- There is no need to invert and;
- Multiplication depth remains reasonable.

The first condition stems from the (conjectured) absence of efficient tropical inversion. The second is due to the fact that the degree and the coefficients of the involved polynomials grows as we keep \otimes ing.

At a first glance those conditions do not seem to apply to schemes such as Diffie-Hellman. Fortunately, two interesting observations may still salvage the situation.

First we observe that if $P_i \in \mathcal{P}_{r,d}$ then:

$$\bigotimes_{i=0}^{\ell-1} P_i \in \mathcal{P}_{\ell r, \ell d}$$

It follows that even if multiplication depth is huge, e.g. in a tropical Diffie-Helman with 1024-bit exponents, coefficients will be large but manageable. The degree of the resulting polynomial is however problematic as, in the example given, we would end-up with polynomials of degree $(\ell d)^2 = 2^{2048} d^2$.

The workaround may consist in reducing the resulting polynomials modulo x^q , i.e. working in $S[t]/(t^q)$ chopping all terms whose degree exceeds q . e.g., one could consider $q = 10d$. Working modulo polynomials more complex than x^q (e.g. $S[t]/(t^q \pm 1)$) is yet another option and has the advantage of recycling the “most significant” information of the polynomials while preserving size. Such approaches would allow tropicalizing cryptosystems with high multiplication depth such as Diffie-Hellman. This rough and general blueprint requires a deeper analysis because the coefficients of the polynomial G^x in $S[x]$ are, in essence, very close to a small constant times x . Reducing each coefficient modulo some small prime modulus e seems to avoid this problem but might create others.

The case of ElGamal variants where inversion is not required is interesting. Such variants exist (e.g. EG I.3 or EG I.4 in [Horster, 1994]) but now the s part of the signature must be given in \mathbb{Z} which might be vulnerable and deserves further investigations and/or new countermeasures.

²¹Be it signature, verification, encryption or decryption.

5.5 Implementation

We implemented all the algorithms mentioned above, as well as CGS, to obtain rough estimates of the associated overheads.

5.5.1 Timings.

The timings provided below were obtained as the median of 100 runs on an 8-th generation Intel Core i7, after a 3 second warmup. All algorithms are implemented in Rust and compiled with `rustc` version 1.77.

Tropical Diffie–Hellman. Key-exchange is performed in 514ms for $(r, d, q) = (32, 32, 64)$.

CGS. Key generation is performed in 318 μ s, signature is performed in 2.15ms and verification is performed in 7.49ms for $(r, d) = (127, 150)$.

FS-CGS. Key generation is performed in 318 μ s, signature is performed in 630ms and verification is performed in 647ms for $(r, d, \tau) = (127, 150, 128)$.

Tropical FS. Key generation is performed in 310 μ s, commit phase takes 304 μ s, response phase takes 160ns and verification is performed in 312 μ s for $(r, d) = (127, 150)$.

5.5.2 Additional details.

We use SHAKE128 (instead of SHA-512) for fast hashing to polynomials. This, the use of a compiled language, and a more straightforward implementation of tropical operations result in a 20–25 \times speedup compared to the reference implementation of [Chen, 2023], despite the workstation being much less powerful. FS-CGS is the “vanilla” version. Tropical Diffie–Hellman uses simple truncation above degree q and does not validate parameters (this would not have a large impact on timings).

5.5.3 Comments.

Assuming the chosen parameters provide the expected security level, CGS, FS-CGS, and tropical Diffie–Hellman seem to be far too slow for practical deployment. Tropical Fiat–Shamir identification is 100 times slower than “standard” Fiat–Shamir. The Rust code is available from the authors upon request.

Chapter 5

Conclusions and Perspectives

1 Conclusions

Within this thesis, we have uncovered the profound impact of mathematical ingenuity in addressing real-world challenges. Our journey through the research findings and contributions attempts to seamlessly connect the dots:

We initiated our exploration with a deep dive into optimal pool testing, harnessing mathematical principles to efficiently identify negative and positive samples. This approach not only extends to exact solutions for smaller sample sizes but also offers heuristic algorithms for larger sets. Although the quest for a polynomial-time algorithm for larger sets remains an open question, the simplicity of implementation once a metaprocedure is established is promising. This has significant implications for fields like medicine and engineering testing, where optimizing performance by addressing false positives and false negatives is crucial. Our exploration then led us to adaptive pool testing, where we discovered the potential to accelerate result acquisition and dynamically influence the sequence of results. This dynamic approach reshapes testing procedures, enhancing their efficiency. Remaining in the realm of solutions to issues related to Covid-19 tests, we introduced an innovative method to safeguard the privacy of DNA information within patient biological samples during processing. By addressing critical data security and confidentiality concerns, this mathematical solution holds the potential to improve healthcare practices.

Transitioning to mathematical conjecture validation, our research demonstrated the efficiency of automating the validation of numerous conjectures. This approach eliminates the need for laborious individual machine proofs. Furthermore, we harnessed the power of pattern matching to uncover fresh mathematical conjectures, thereby showing the usefulness of pattern matching in conjecture detection and mathematical exploration.

Lastly, our research spotlighted a practical concern within zero-knowledge protocols. It became evident that even brief challenges of 20 or 40 bits can compromise the prover's deniability in practical contexts. To address this issue, we recommended a solution: the verification of random challenge selection. This clean and effective resolution can be readily implemented in scenarios where deniability is paramount.

While spreading over very diverse sub-areas this thesis illustrates one main narrative: the joy of discovering how mathematical innovation plays a pivotal role in addressing real-world challenges.

2 Perspectives and Further Research

The following questions can be used as a basis for further research subsequent to our work:

2.1 Pool Testing Research Challenges

- What is the optimal pool size and composition to balance sensitivity and cost-effectiveness in different population settings?
- What statistical approaches can be developed to accurately estimate infection prevalence and individual testing probabilities within pooled samples?
- How can dynamic pooling strategies be implemented to adjust pool sizes and compositions in real-time based on infection prevalence?
- What are the challenges and solutions for extending pool testing to complex sample types, such as wastewater and environmental samples? What if the test reacts in proportion to a viral load according to some probability distribution and is not binary?
- How can pool testing data be integrated into epidemiological models to assess its impact on disease control and surveillance?
- Create adaptive pooling algorithms that can dynamically adjust pool sizes based on the observed prevalence of the disease. This requires real-time decision-making to balance sensitivity and efficiency.
- Investigate matrix sampling techniques, where samples are grouped into matrices to enable efficient testing. Develop mathematical approaches to minimize the number of tests required when using matrix-based pooling.
- Address the issue of overlapping pools, where a single sample may be part of multiple pools. Develop mathematical methods to resolve potential interference and accurately identify positive cases.
- Analyze the asymptotic properties of pool testing procedures as the number of samples and pool sizes become large. Investigate limit theorems, consistency, and efficiency of estimators in this context.
- Determine the optimal allocation of testing resources, including the number of samples to pool, the size of pools, and the frequency of testing, to minimize costs while meeting desired sensitivity levels.

2.2 Experimental Mathematics Research Challenges

Evidently, the following list covers only the open questions that stem from our thesis work. This is only a very small part of the extremely wide realm of experimental mathematics.

- Prove the relations detected computationally that we list in this thesis. If possible provide a way to automate the generation of such proofs.
- What happens in the Balkans for even j values?
- What happens in Inostranstvo?

- Reverse the continued fraction generation process to match a target constant as mentioned in section 1.5.
- Investigate the existence of continued fractions having an a_n of the form

$$a_n = (v + \kappa n)(v + 1 + \kappa n)(v + 2 + \kappa n)(v + 3 + \kappa n)$$

As mentioned in sub-section 2.4.2

- What is the relation between OEIS sequence A006309 and $f(x)$ mentioned in sub-section 2.7.2? Why are the elements 12803, 14615 and 11537 missing?

2.3 Information Security Research Challenges

In addition to the above, the following two challenges add-up to the list of open question raised by this thesis:

- Is there a way to shorten signatures by more than 3ℓ bits under an $O(2^\ell)$ work constraint by both parties?
- Is there a way to extend the shortening strategies explored in this thesis to non-randomized signature schemes such as Gui?

3 Thesis Defense Reports



Rapport sur la Thèse de M. Ofer Stav-Ifrach : "Tests par lots rapides et privés et contributions aux mathématiques expérimentales"

Par Naila Hayek, Professeur de Mathématiques Appliquées à l'Université Paris Panthéon-Assas. 12 place du Panthéon 75005

La thèse de M. Ofer Stav-Ifrach couvre un spectre très vaste de sujets, allant de la recherche sur la Covid-19 à l'expérimentation mathématique et à la sécurité de l'information. Ce rapport présente les principales contributions du candidat et souligne la qualité de son travail.

1. Recherches liées à la Covid-19 :

La première partie de la thèse explore la recherche liée à la Covid-19 avec une approche innovante. L'auteur se penche sur l'optimisation des tests par lots de dépistage avec des informations a priori, incorporant des notions mathématiques et combinatoires complexes. Il généralise ces tests en introduisant également une notion d'urgence. Il examine également la préservation de la confidentialité des données ADN lors de la détection à grande échelle du virus. L'adaptation réussie de concepts classiques de cryptographie témoigne de l'originalité de l'auteur.

2. Mathématiques expérimentales :

La deuxième partie de la thèse s'intéresse à la "Machine Ramanujan" d'un point de vue mathématique et informatique. L'auteur apporte une explication de la raison pour laquelle la Machine Ramanujan arrive à détecter certaines relations. Il réalise des expériences de reconnaissance de motifs sur des expressions mathématiques, proposant des conjectures vérifiées numériquement avec une grande précision. L'étude approfondie de la "fraction continue des Balkans" est impressionnante, démontrant la capacité de l'intelligence artificielle à générer de nouvelles conjectures.

3. Contributions pratiques à la sécurité de l'information :

La dernière partie de la thèse aborde des questions de sécurité de l'information en s'intéressant aux "attaques par formules invisibles" présentées à la conférence "BlackHat". Cette attaque consiste à utiliser des fonctions de visualisation intelligente afin d'insérer du code malveillant dans des programmes. L'auteur explore également des aspects pratiques tels que l'authentification des médicaments via des codes QR et des signatures numériques compactes.

Évaluation :

La thèse de M. Ofer Stav-Ifrach offre une perspective interdisciplinaire très riche, couvrant des domaines allant de la biologie aux mathématiques expérimentales et à la sécurité de l'information. La diversité thématique, la profondeur des résultats obtenus, et la clarté de l'écriture témoignent d'une grande maîtrise. L'auteur se révèle être un scientifique ayant de grandes compétences et une boîte à outils scientifique impressionnante.

Pour toutes ces raisons, je recommande chaleureusement la défense de cette thèse. M. Ofer Stav-Ifrach a non seulement démontré une compréhension approfondie des sujets abordés mais aussi une capacité rare à aborder des problèmes variés avec clarté et précision. Je suis convaincue qu'il continuera à contribuer significativement à la recherche académique ou industrielle en tant que futur enseignant-chercheur ou chercheur.

Paris le 22 Janvier 2024

A handwritten signature in blue ink, appearing to read 'Wang' followed by a stylized flourish.

Paris, le 22 décembre 2023

Objet : Mémoire de doctorat d'Ofer Yifrach-Stav

Madame, Monsieur,

Le traitement efficace et la sécurisation des données joue un rôle prépondérant dans la mise en œuvre pratique de nombreuses applications. Ainsi, lorsqu'un test sur des données est coûteux ou prend du temps, une approche connue est de recourir aux tests par lots. Plusieurs échantillons sont regroupés dans un même lot et le test est effectué sur ce lot au prix d'un unique test. Le but est de minimiser le nombre total de tests tout en garantissant individuellement l'exactitude du test pour chacun des échantillons. La thèse d'Ofer Yifrach-Stav s'intéresse aux tests par lots dans le cadre du dépistage de la Covid-19. Ainsi, en faisant des hypothèses a priori sur la probabilité qu'un patient est porteur ou non du virus, l'auteur développe des méthodes quasi-optimales pour détecter la présence ou non de charge virale. Une problématique connexe dans le traitement de données, en particulier pour les données médicales, est de préserver leur caractère privé. Dans ce contexte, l'auteur adapte les notions d'indistingabilité employées dans la définition des algorithmes de chiffrement afin de garantir la confidentialité des données traitées lors des tests de dépistage.

L'avènement des ordinateurs et des logiciels de calcul scientifique de type Mathematica a vu l'émergence d'une branche des mathématiques connue sous le nom de «mathématiques expérimentales». La thèse d'Ofer Yifrach-Stav s'intéresse ici à la machine de Ramanujan. Cette machine vise à découvrir de nouvelles formules mathématiques. Elle a ainsi permis de développer de nombreuses conjectures mathématiques, vérifiées numériquement avec une grande précision. Dans cette partie, l'auteur tire partie de la puissance des techniques d'apprentissage, en particulier des méthodes de descente de gradient, pour expliquer et généraliser les conjectures établies

par la machine de Ramanujan. L'auteur rapporte également une multitude de nouvelles conjectures. On note les jolies formules obtenues pour une variété de fractions continues.

Dans une dernière partie, la thèse d'Ofer Yifrach-Stav traite de la sécurité de l'information. L'auteur présente une série de contributions concrètes, à savoir des attaques de formules invisibles dans Mathematica, des motifs dans le calcul de carrés modulo un nombre de Mersenne, le risque lié aux défis choisis dans les protocoles de preuve à divulgation nulle, les codes QR pour l'authentification des médicaments et les signatures numériques courtes.

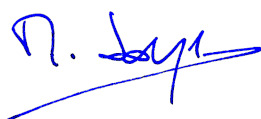
Le manuscrit de thèse, par ailleurs très bien écrit, est organisé suivant les trois parties décrites plus haut : les travaux liés à la Covid-19, les mathématiques expérimentales et la sécurité de l'information. Le manuscrit inclut également un chapitre concluant les différentes contributions et offrant des perspectives de futures recherches pour chacune des trois parties. Certains résultats obtenus ont donné lieu à des publications dans des conférences internationales avec comité de relecture. D'autres résultats sont disponibles sous la forme de preprints. Le format original de la thèse est en partie dû aux circonstances exceptionnelles liées à la pandémie. Les recherches du candidat sur la Covid-19 n'étaient évidemment pas prévues initialement. Il y a néanmoins une unité dans l'approche d'aborder les problèmes et de les solutionner. On retrouve également une unité dans la thèse elle-même. La thématique de façon générale peut se résumer à l'étude du traitement de l'information, de sa sécurisation et de sa généralisation.

Ce travail met en évidence la capacité du candidat à travailler sur des sujets très variés et pluridisciplinaires. Ses travaux démontrent une maturité scientifique certaine. Il emprunte des outils et concepts de différents domaines des mathématiques et du traitement de l'information au sens large et les applique avec succès.

Ainsi, au vu des contributions apportées par ce mémoire, j'émet sans réserve un avis favorable à la soutenance.

Je vous prie d'agréer, Madame, Monsieur, mes salutations les meilleures.

Cordialement,



Marc Joye, PhD, HDR
Chief Scientist @ Zama

Nom et prénom du doctorant : YIFRACH-STAV Ofer

Date de la soutenance : 28 février 2024

Président du Jury : PETER YA RYAN

Le 28/02/2024, M. Ofer Yifrach-Stav a soutenu sa thèse intitulée « *Tests par lots rapides et privés et contributions aux mathématiques expérimentales* ». Le jury a trouvé son exposé très clair et précis. Le jury a été conquis par la passion émanant de ses explications.

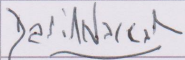
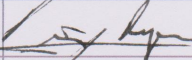

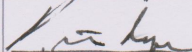
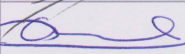
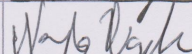
M. Ofer Yifrach-Stav a expliqué de manière accessible et pédagogique des sujets difficiles et complexes et a illustré son propos remarquablement étape par étape.

Les travaux couvrent un spectre exceptionnellement large de disciplines scientifiques dont les mathématiques, les statistiques, la combinatoire, la programmation, la sécurité informatique et la bio-informatique. Les résultats trouvent des applications pratiques en sécurité informatique et en recherche épidémiologique (Covid-19).

Le candidat a fait preuve d'une rare maîtrise de nombreux outils et techniques empruntés avec succès à des domaines très divers. Les résultats sont d'une grande originalité et d'une excellente qualité scientifique. Le jury est convaincu que le candidat sera un excellent chercheur ou enseignant-chercheur, tant dans le monde académique qu'industriel.

Le candidat a prononcé le serment du docteur (Art. 16 de l'Arrêté du 25 mai 2016 modifié par l'Arrêté du 26 août 2022) devant le jury.

Pour toutes ces raisons, le jury à l'unanimité, confère à M. Ofer Yifrach-Stav le « *titre de docteur de l'Université PSL, thèse préparée à l'Ecole normale supérieure* ».

Prénom et Nom	Signature	Prénom et Nom	Signature
David NACCACHE		Jean-Jacques QUISQUATER	
Marc JOYE		Peter Y A RYAN	
Robert WEIL		Naïla HAYEK	

List of Publications

Peer-Reviewed Journal Articles

- [Tomkins, 2007] Oren Tomkins, Ofer Friedman, Sebastian Ivens, Clemens Reiffurth, Sebastian Major, Jens Peter Dreier, Uwe Heinemann, and Alon Friedman. “Blood–Brain Barrier Disruption Results in Delayed Functional and structural Alterations in the Rat Neocortex”. *Neurobiology of disease* 25.2 (2007), pp. 367–377.

Electronic Pre-Prints

- [Beunardeau, 2020] Marc Beunardeau, Éric Brier, Noémie Cartier, Aisling Connolly, Nathanaël Courant, Rémi Géraud-Stewart, David Naccache, and Ofer Yifrach-Stav. “Optimal Covid-19 Pool Testing with *A Priori* Information”. *arXiv preprint arXiv:2005.02940* (2020).
- [Brier, 2021] Éric Brier, Megi Dervishi, Rémi Géraud-Stewart, David Naccache, and Ofer Yifrach-Stav. “Near-Optimal Pool Testing Under Urgency Constraints”. *arXiv preprint arXiv:2106.10971* (2021).
- [Brier, 2022] Éric Brier, David Naccache, and Ofer Yifrach-Stav. “A Note on the Ramanujan Machine”. *arXiv preprint arXiv: 2211.01058* (2022).
- [Géraud-Stewart, 2023] Rémi Géraud-Stewart, David Naccache, and Ofer Yifrach-Stav. “Fiat-Shamir Goes Tropical”. *Cryptology ePrint Archive, Paper 2023/1954* (2023).
- [Hollenstein, 2020] Marcel Hollenstein, David Naccache, Peter B. Rønne, Peter Y. A. Ryan, Robert Weil, and Ofer Yifrach-Stav. “Preservation of DNA Privacy During the Large Scale Detection of Covid-19”. *arXiv preprint arXiv:2007.09085* (2020).
- [Jainsky, 2023] Julien S. Jainsky, David Naccache, Bassem Ouni, and Ofer Yifrach-Stav. “Authenticating Medications with QR-Codes and Compact Digital Signatures”. *IACR Cryptol. ePrint Arch.* (2023), p. 1817.
- [Naccache, 2022a] David Naccache and Ofer Yifrach-Stav. “A Conjecture From a Failed Cryptanalysis”. *IACR Cryptol. ePrint Arch.* (2022), p. 1273.
- [Naccache, 2022b] David Naccache and Ofer Yifrach-Stav. “Invisible Formula Attacks”. *IACR Cryptol. ePrint Arch.* (2022), p. 1110.
- [Naccache, 2022d] David Naccache and Ofer Yifrach-Stav. “On Squaring Modulo Mersenne Numbers”. *IACR Cryptol. ePrint Arch.* (2022), p. 1197.
- [Naccache, 2022e] David Naccache and Ofer Yifrach-Stav. “Pattern Recognition Experiments on Mathematical Expressions”. *arXiv preprint arXiv:2301.01624* (2022).

Publications by Ofer Yifrach-Stav are under the name Ofer Friedman, if published before 2010.

- [Naccache, 2023a] David Naccache and Ofer Yifrach-Stav. “A Note on Moments of Random Multiplicative Functions and Truncated Characteristic Polynomials”. *HAL* 04089572 (2023).
- [Naccache, 2023b] David Naccache and Ofer Yifrach-Stav. “Continued Fractions With Numerator $(v + 1 + n)(v + 2 + n)(v + 3 + n)(v + 4 + n)$ ”. *HAL* 04092469 (2023).
- [Naccache, 2023c] David Naccache and Ofer Yifrach-Stav. “Explicit Formulae for Some Polynomial Continued Fractions”. *HAL* 04105623 (2023).
- [Naccache, 2023f] David Naccache and Ofer Yifrach-Stav. “On The Practical Advantage of Committing Challenges in Zero-Knowledge Protocols”. *Cryptography ePrint Archive, Paper 2023/1961* (2023).
- [Naccache, 2023g] David Naccache and Ofer Yifrach-Stav. “The Balkans Continued Fraction”. *arXiv preprint arXiv:2308.06291* (2023).
- [Yifrach-Stav, 2023] Ofer Yifrach-Stav. “Can medical devices be used to rob banks?” *IPA - International Pharmaceutical Academy* (2023).

Peer-reviewed International Conferences

- [Friedman, 2006] Ofer Friedman. “Anatomical Degeneration and Functional Deterioration in the Rat Epileptic Cortex”. *Epilepsia*. Vol. 47. Blackwell Publishing 9600 Garsington RD, Oxford OX4 2DQ, OXON, England. 2006, pp. 114–114.
- [Friedman, 2005b] Ofer Friedman, Oren Tomkins, Ofer Prager, Hilit Lis, Sebastian Ivens, Sebastian Major, Jens Dreier, and Alon Friedman. “Delayed Neurological and Behavioral Deterioration Following Blood-Brain Barrier Disruption in the Rat Motor Cortex”. *Reviews in the Neurosciences*. Vol. 16. Freund & Pettman Publishers Enholmes Hal, Partrington, East Yorkshire HU12 OPR, England. 2005, S22–S22.
- [Naccache, 2022c] David Naccache and Ofer Yifrach-Stav. “Invisible Formula Attacks”. The Black Hat (Las Vegas, USA). 2022.
- [Naccache, 2023d] David Naccache and Ofer Yifrach-Stav. “On Catalan Constant Continued Fractions”. *Codes, Cryptology and Information Security*. Ed. by Said El Hajji, Sihem Mesnager, and El Mamoun Soudi. Cham: Springer Nature Switzerland, 2023, pp. 43–54 (cit. on p. 105).

National Conference and Seminar Presentation

- [Friedman, 2005a] Ofer Friedman. *Behaviour Study in Rats Following Blood Brain Barrier Disruption*. Annual Retreat of the Zoltowski Center for Neuro- science, Mitzpe Ramon, Israel, 2005.

Patents

- [Chamberland, 2020] Guy Chamberland, Charles Campbell, Randy Ringguette, Jenniver Dorothy Bassett, Ofer Yifrach-Stav, Andrien Rackov, and Peter Ford. *Cannabis Compositions and Methods*. Publication number: CA3027876A1, filed: Dec, 2018, published: June, 2020.
- [Kaur, 2021] Harpreet Kaur, Xuejun Liu, Prabin Nepal, Subakar Paramanantham, Monika Garg, Zemin Li, Andy Tjeng, Moises Rodriguez, Laura Jong, Azam Mizrahossein, Ofer Yifrach-Stav, Erin Bassett, Randy Ringuette, Charles Campbell, and Melanie Kelly. *Methods, Processes, and Compositions for Improved Preparation of hu308 and hu433*. Publication number: WO2022061461A1, filed: Sep, 2020, published: Sep, 2021.

Bibliography

- [Aldridge, 2019] Matthew Aldridge, Oliver Johnson, and Jonathan Scarlett. “Group Testing: an Information Theory Perspective”. *arXiv preprint arXiv:1902.06002* (2019) (cit. on pp. 20, 21).
- [Allemann, 2013] Andreas Allemann. “An Efficient Algorithm for Combinatorial Group Testing”. *Information Theory, Combinatorics, and Search Theory*. Springer, 2013, pp. 569–596 (cit. on p. 20).
- [Ambion,] ThermoFisher Scientific Ambion. *Now It’s Easy to Make your RNA Free of Genomic DNA Contamination and Ready for RT-PCR* (cit. on p. 67).
- [Annas, 2004] George Annas. “Genetic Privacy”. *DNA and the Criminal Justice System: The Technology of Justice* (2004), pp. 135–46 (cit. on p. 4).
- [Assad, 2020] Assif Assad, Muzafar Ahmad Wani, and Kusum Deep. “A Comprehensive Strategy to Lower Number of Covid-19 Tests”. *Available at SSRN 3578240* (2020) (cit. on p. 22).
- [Bailey, 1989] David H. Bailey and Howard R. P. Ferguson. “Numerical Results on Relations Between Numerical Constants Using a New Algorithm”. *Mathematics of Computation* 53.188 (1989), pp. 649–656 (cit. on p. 110).
- [Baker, 2020] Scott Baker, Nicholas Bloom, Steven J. Davis, Kyle Kost, Marco Sammon, and Tasaneeya Viratyosin. “The Unprecedented Stock Market Reaction to Covid-19”. *Covid Economics: Vetted and Real-Time Papers* 1.3 (2020) (cit. on p. 19).
- [Bana, 2021] Gergei Bana, Wojciech Jamroga, David Naccache, and Peter Y. A. Ryan. “Convergence Voting: From Pairwise Comparisons to Consensus”. *CoRR* abs/2102.01995 (2021). arXiv: 2102.01995 (cit. on p. 134).
- [Baptista, 2008] Pedro Baptista, Eulália Pereira, Peter Eaton, Gonçalo Doria, Adelaide Miranda, Inês Gomes, Pedro Quaresma, and Ricardo Franco. “Gold Nanoparticles for the Development of Clinical Diagnosis Methods”. *Analytical and bioanalytical chemistry* 391.3 (2008), pp. 943–950 (cit. on p. 67).
- [Barral, 2021] Hadrien Barral, Éric Brier, Rémi Géraud-Stewart, Arthur Léonard, David Naccache, Quentin Vermande, and Samuel Vivien. *Discovering New L-Function Relations Using Algebraic Sieving*. Cryptology ePrint Archive, Paper 2021/1060. 2021 (cit. on p. 120).
- [Bauder, 1983] Don Bauder. *An Anti-Counterfeiting Concept for Currency Systems*. Tech. rep. PTK-11990. Albuquerque, NM: Sandia National Labs, 1983 (cit. on p. 181).
- [BBC, 2020] BBC. *Coronavirus: UK Sent 50,000 Covid-19 Samples to US for Testing*. 2020 (cit. on p. 62).
- [Beaudevin, 2021] Claire Beaudevin, Luc Berlivet, Soraya Boudia, Catherine Bourgain, Maurice Cassier, Jean-Paul Gaudillère, and Ilana Löwy. “‘Test, Test, Test!’: Scarcity, Tinkering, and Testing Policy Early in the Covid-19 Epidemic in France”. *Medicine Anthropology Theory* 8.2 (2021), pp. 1–31 (cit. on p. 2).
- [Bellare, 1998] Mihir Bellare and Phillip Rogaway. *PSS: Provably Secure Encoding Method for Digital Signatures*. 1998 (cit. on p. 169).
- [Ben-Ami, 2020] Roni Ben-Ami, Agnes Klochendler, Matan Seidel, Tal Sido, Ori Gurel-Gurevich, Moran Yassour, Eran Meshorer, Gil Benedek, Irit Fogel, Esther Oiknine-Djian, Asaf Gertler, Zeev Rotstein, Bruno Lavi, Yuval Dor, Dana G. Wolf, Maayan Salton, and Yotam Drier. “Pooled RNA Extraction and PCR Assay for Efficient SARS-COV-2 Detection”. *medRxiv* (2020). eprint: <https://www.medrxiv.org/content/early/2020/04/22/2020.04.17.20069062.full.pdf> (cit. on p. 22).
- [Benhamouda, 2017] Fabrice Benhamouda, Houda Ferradi, Rémi Géraud, and David Naccache. *Non-Interactive Provably Secure Attestations for Arbitrary RSA Prime Generation Algorithms*. Cryptology ePrint Archive, Paper 2017/640. 2017 (cit. on p. 193).
- [Beunardeau, 2020] Marc Beunardeau, Éric Brier, Noémie Cartier, Aisling Connolly, Nathanaël Courant, Rémi Géraud-Stewart, David Naccache, and Ofer Yifrach-Stav. “Optimal Covid-19 Pool Testing with *A Priori* Information”. *arXiv preprint arXiv:2005.02940* (2020).
- [Bierlaire, 2017] Damien Bierlaire, Sylvie Mauguin, Julien Brout, and Didier Musso. “Zika Virus and Blood Transfusion: the Experience of French Polynesia”. *Transfusion* 57.3pt2 (2017), pp. 729–733 (cit. on pp. 2, 21).

- [Bilder, 2010] Christopher R. Bilder, Joshua M. Tebbs, and Peng Chen. “Informative Retesting”. *Journal of the American Statistical Association* 105.491 (2010), pp. 942–955 (cit. on p. 20).
- [Biolabs, 2020] New-England Biolabs. *A Typical DNase I Reaction Protocol (M0303)*. 2020 (cit. on p. 67).
- [Black, 2012] Michael S. Black, Christopher R. Bilder, and Joshua M. Tebbs. “Group Testing in Heterogeneous Populations by Using Halving Algorithms”. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 61.2 (2012), pp. 277–290 (cit. on p. 20).
- [Board, 1998] Space Studies Board, National Research Council, et al. *Privacy Issues in Biomedical and Clinical Research. In: A strategy for Research in Space Biology and Medicine in the New century*. National Academies Press, 1998 (cit. on p. 62).
- [Borisov, 2002] Nikita Borisov, Monica Chew, Robert Johnson, and David Wagner. “Multiplicative Differentials”. *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*. Vol. 2365. Lecture Notes in Computer Science. Springer, 2002, pp. 17–33 (cit. on p. 172).
- [Brier, 2021] Éric Brier, Megi Dervishi, Rémi Géraud-Stewart, David Naccache, and Ofer Yifrach-Stav. “Near-Optimal Pool Testing Under Urgency Constraints”. *arXiv preprint arXiv:2106.10971* (2021).
- [Brier, 2022] Éric Brier, David Naccache, and Ofer Yifrach-Stav. “A Note on the Ramanujan Machine”. *arXiv preprint arXiv : 2211.01058* (2022).
- [Brown, 2023] Daniel R. L. Brown and Chris Monico. *More forging (and patching) of tropical signatures*. Cryptology ePrint Archive, Paper 2023/1837. 2023 (cit. on pp. 18, 189).
- [Cambon-Thomsen, 2004] Anne Cambon-Thomsen. “The Social and Ethical Issues of Post-Genomic Human Biobanks”. *Nature Reviews Genetics* 5.11 (2004), pp. 866–873 (cit. on p. 62).
- [Cannataci, 2016] Joseph A. Cannataci. “Report of the Special Rapporteur on the Right to Privacy”. *Human Rights Council* (2016) (cit. on p. 62).
- [Capocasa, 2016] Marco Capocasa, Paolo Anagnostou, Flavio D’Abramo, Giulia Matteucci, Valentina Dominici, Giovanni Destro Bisol, and Fabrizio Rufo. “Samples and Data Accessibility in Research Biobanks: an Explorative Survey”. *PeerJ* 4 (2016), e1613 (cit. on p. 61).
- [CBC, 2016] CBC. *Alberta Health Services Notifies Almost 13,000 patients of Privacy Breach*. 2016 (cit. on p. 62).
- [Chamberland, 2020] Guy Chamberland, Charles Campbell, Randy Ringgnette, Jenniver Dorothy Bassett, Ofer Yifrach-Stav, Andrien Rackov, and Peter Ford. *Cannabis Compositions and Methods*. Publication number: CA3027876A1, filed: Dec, 2018, published: June, 2020.
- [Chan, 2013] Hei-Chi Chan. “Golden Ratio and a Ramanujan-Type Integral”. *Axioms* 2.1 (2013), pp. 58–66 (cit. on pp. 89, 91).
- [Chang-won, 2020] Lim Chang-won. “Verified ‘Sample Pooling’ Introduced to Prevent Herd Infection in S. Korea”. *Aju Business Daily* (2020) (cit. on p. 22).
- [Chen, 2023] Jiale Chen, Dima Grigoriev, and Vladimir Shpilrain. *Tropical cryptography III: digital signatures*. Cryptology ePrint Archive, Paper 2023/1475. 2023 (cit. on pp. 18, 189, 194).
- [Clark, 2015] Fiona Clark. “Rise in Online Pharmacies Sees Counterfeit Drugs Go Global”. *The Lancet* 386.10001 (2015), pp. 1327–1328 (cit. on p. 181).
- [Cloître, 2004] Benoît Cloître. *A101269*. oeis.org/A101269. 2004 (cit. on p. 118).
- [Cloître,] Benoît Cloître. *A BBP Formula for π^2 in Golden Base* (cit. on p. 96).
- [Cohen, 2022] Henri Cohen. *Elementary Continued Fractions for Linear Combinations of ζ and L Values*. 2022. arXiv: 2212.01095 [math.NT] (cit. on p. 106).
- [Coja-Oghlan, 2019] Amin Coja-Oghlan, Oliver Gebhard, Max Hahn-Klimroth, and Philipp Loick. “Information-Theoretic and Algorithmic Thresholds for Group Testing”. *arXiv preprint arXiv:1902.02202* (2019) (cit. on p. 21).
- [Coppersmith, 1996] Don Coppersmith. “Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known”. *Advances in Cryptology - EUROCRYPT ’96*. Ed. by Ueli Maurer. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 178–189 (cit. on p. 169).
- [Cripps, 2020] Karla Cripps. “Airline Passengers Undergo Covid-19 Blood Tests Before Boarding”. *News4JAX* (2020) (cit. on p. 62).
- [Cucinotta, 2020] Domenico Cucinotta and Maurizio Vanelli. “WHO Declares Covid-19 a Pandemic”. *Acta Bio Medica: Atenei Parmensis* 91.1 (2020), p. 157 (cit. on p. 1).
- [Currie, 2004] Marian J. Currie, Michelle McNiven, Tracey Yee, Ursula Schiemer, and Francis J. Bowden. “Pooling of Clinical Specimens Prior to Testing for Chlamydia Trachomatis by PCR is Accurate and Cost Saving”. *Journal of clinical microbiology* 42.10 (2004), pp. 4866–4867 (cit. on p. 21).
- [Damgård, 2010] Ivan Damgård. *On Σ Protocols*. 2010 (cit. on p. 173).

- [David, 2021] Nadav Ben David, Guy Nimri, Uri Mendlovic, Yahel Manor, and Ido Kaminer. “On the Connection Between Irrationality Measures and Polynomial Continued Fractions”. *arXiv preprint arXiv:2111.04468* (2021) (cit. on pp. 71, 84).
- [David, 2023] Ofir David. *The Conservative Matrix Field*. 2023. arXiv: 2303.09318 [math.GM] (cit. on p. 117).
- [Dorfman, 1943] Robert Dorfman. “The Detection of Defective Members of Large Populations”. *The Annals of Mathematical Statistics* 14.4 (1943), pp. 436–440 (cit. on pp. 2, 20).
- [Du, 2000] Dingzhu Du, Frank K. Hwang, and Frank Hwang. *Combinatorial Group Testing and its Applications*. Vol. 12. World Scientific, 2000 (cit. on p. 20).
- [Dwork, 1992] Cynthia Dwork and Moni Naor. “Pricing via Processing or Combatting Junk Mail”. *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*. 1992, pp. 139–147.
- [Eberhardt, 2020] Jens Niklas Eberhardt, Nikolas Peter Breuckmann, and Christiane Sigrig Eberhardt. “Multi-Stage Group Testing Improves Efficiency of Large-Scale Covid-19 Screening”. *Journal of Clinical Virology* (2020), p. 104382 (cit. on p. 22).
- [ElGamal, 1985] Taher ElGamal. “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”. *Advances in Cryptology—CRYPTO'85* (1985), pp. 10–18 (cit. on p. 174).
- [Elimelech, 2023] Rotem Elimelech, Ofir David, Carlos De la Cruz Mengual, Rotem Kalisch, Wolfgang Berndt, Michael Shalyt, Mark Silberstein, Yaron Hadad, and Ido Kaminer. *Algorithm-Assisted Discovery of an Intrinsic Order Among Mathematical Constants*. 2023. arXiv: 2308.11829 [cs.AI] (cit. on p. 5).
- [Ellis, 2020] Blake Ellis, Melanie Hicken, and Ashley Fantz. “Coroners Worry Covid-19 Test Shortages Could Lead to Uncounted Deaths”. *CNN Investigates* (2020) (cit. on p. 19).
- [Emanuel, 2020] Ezekiel J Emanuel, Govind Persad, Ross Upshur, Beatriz Thome, Michael Parker, Aaron Glickman, Cathy Zhang, Connor Boyle, Maxwell Smith, and James P Phillips. *Fair Allocation of Scarce Medical Resources in the Time of Covid-19*. 2020 (cit. on p. 2).
- [Emmanuel, 1988] Jean C. Emmanuel, Mary T Bassett, Heather J Smith, and J. A. Jacobs. “Pooling of Sera for Human Immunodeficiency Virus (HIV) Testing: an Economical Method for use in Developing Countries.” *Journal of Clinical Pathology* 41.5 (1988), pp. 582–585 (cit. on p. 21).
- [Erdman, 2020] Shelby Lin Erdman. “Public, Private Health Labs May Never be Able to Meet Demand for Coronavirus Testing over Supply Chain Shortages”. *CNN Health* (2020) (cit. on p. 22).
- [Farfan, 2020] Mauricio J. Farfan, Juan P. Torres, Miguel Oryan, Mauricio Olivares, Pablo Gallardo, and Carolina Salas. “Optimizing RT-PCR Detection of SARS-COV-2 for Developing Countries Using Pool Testing”. *medRxiv* (2020). eprint: <https://www.medrxiv.org/content/early/2020/04/17/2020.04.15.20067199.full.pdf> (cit. on p. 22).
- [FDA, 2023] FDA. *Covid-19 Test Basics*. U.S Food & Drug Administration. 2023 (cit. on p. 19).
- [Feige, 1988] Uriel Feige, Amos Fiat, and Adi Shamir. “Zero-Knowledge Proofs of Identity”. *J. Cryptology* 1.2 (1988), pp. 77–94 (cit. on p. 174).
- [Feng, 2017] Emily Feng. “China Authorities Mandated to Collect DNA from Xinjiang Residents”. *Financial Times* (2017) (cit. on p. 62).
- [Ferguson, 1988] Howard R. P. Ferguson. “PSOS: A New Integral Relation Finding Algorithm Involving Partial Sums of Squares and No Square Roots”. *Abstracts of Papers Presented to the American Mathematical Society* 9.56 (1988), 88T-11–75, 214 (cit. on p. 110).
- [Ferguson, 1999] Howard R. P. Ferguson, David H. Bailey, and Steve Arno. “Analysis of PSLQ, an Integer Relation Finding Algorithm”. *Mathematics of Computation* 68.227 (1999), pp. 351–369 (cit. on p. 110).
- [Ferradi, 2015] Houda Ferradi, Rémi Géraud, Diana Maimuř, David Naccache, and Amaury de Wargny. *Regulating the Pace of von Neumann Correctors*. Cryptology ePrint Archive, Paper 2015/849. 2015 (cit. on p. 186).
- [Ferradi, 2016] Houda Ferradi, Rémi Géraud, and David Naccache. *Slow Motion Zero Knowledge Identifying with Colliding Commitments*. Cryptology ePrint Archive, Paper 2016/399. 2016 (cit. on p. 175).
- [Fiat, 1987] Amos Fiat and Adi Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. *Proceedings on Advances in Cryptology—CRYPTO '86*. Santa Barbara, California, USA: Springer-Verlag, 1987, pp. 186–194 (cit. on p. 190).
- [Flórez, 2019] Rigoberto Flórez, Robinson A. Higuaita, and Antara Mukherjee. *Characterization of the Strong Divisibility Property for Generalized Fibonacci Polynomials*. 2019. arXiv: 1701.06722 [math.NT] (cit. on p. 120).
- [Forensic-Science-Regulator, 2018] UK Government Publications Forensic-Science-Regulator. *DNA Mixture Interpretation, FSR-G-222*. 2018 (cit. on p. 66).

- [Friedman, 2005a] Ofer Friedman. *Behaviour Study in Rats Following Blood Brain Barrier Disruption*. Annual Retreat of the Zoltowski Center for Neuro- science, Mitzpe Ramon, Israel, 2005.
- [Friedman, 2006] Ofer Friedman. “Anatomical Degeneration and Functional Deterioration in the Rat Epileptic Cortex”. *Epilepsia*. Vol. 47. Blackwell Publishing 9600 Garsington RD, Oxford OX4 2DQ, OXON, England. 2006, pp. 114–114.
- [Friedman, 2005b] Ofer Friedman, Oren Tomkins, Ofer Prager, Hilit Lis, Sebastian Ivens, Sebastian Major, Jens Dreier, and Alon Friedman. “Delayed Neurological and Behavioral Deterioration Following Blood-Brain Barrier Disruption in the Rat Motor Cortex”. *Reviews in the Neurosciences*. Vol. 16. Freund & Pettman Publishers Enholmes Hal, Partrington, East Yorkshire HU12 OPR, England. 2005, S22–S22.
- [Géraud-Stewart, 2023] Rémi Géraud-Stewart, David Naccache, and Ofer Yifrach-Stav. “Fiat-Shamir Goes Tropical”. *Cryptology ePrint Archive, Paper 2023/1954* (2023).
- [Ghebreyesus, 2020] Tedros Adhanom Ghebreyesus. *Opening Remarks at the Media Briefing on Covid-19 - 16 March 2020*. 2020 (cit. on p. 22).
- [Girault, 1990] Marc Girault. “An Identity-based Identification Scheme Based on Discrete Logarithms Modulo a Composite Number”. *Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*. 1990, pp. 481–486 (cit. on p. 174).
- [Girault, 1994] Marc Girault and Jacques Stern. “On the Length of Cryptographic Hash-Values Used in Identification Schemes”. *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*. 1994, pp. 202–215 (cit. on p. 175).
- [Goldreich, 1991] Oded Goldreich, Silvio Micali, and Avi Wigderson. “Proofs that Yield Nothing But Their Validity for All Languages in NP Have Zero-Knowledge Proof Systems”. *J. ACM* 38.3 (1991), pp. 691–729 (cit. on p. 174).
- [Goldwasser, 1985] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. “The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)”. *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*. 1985, pp. 291–304 (cit. on p. 173).
- [Gollier, 2020] Christian Gollier and Olivier Gossner. “Group Testing Against Covid-19”. *Covid Economics* 1.2 (2020), pp. 32–42 (cit. on p. 22).
- [Greenbaum, 2011] Dov Greenbaum, Andrea Sboner, Xinmeng Jasmine Mu, and Mark Gerstein. “Genomics and Privacy: Implications of the New Reality of Closed Data for the Field”. *PLoS Computational Biology* 7.12 (2011), e1002278 (cit. on p. 4).
- [Haas, 2017] Benjamin Haas. “Chinese Authorities Collecting DNA from all Residents of Xinjiang”. *The Guardian* (2017) (cit. on p. 62).
- [Hahn, 2020] Stephen M. Hahn. “Coronavirus (Covid-19e) Update: Serological Test Validation and Education Efforts”. *FDA Statement* (2020) (cit. on pp. 19, 20).
- [Harper, 2020] Adam J. Harper. “Moments of Random Multiplicative Functions, I: Low Moments, Better than Square Root Cancellation, and Multiplicative Chaos”. *Forum of Mathematics, Pi* 8 (2020), e1 (cit. on pp. 14, 99).
- [Håstad, 1986] Johann. Håstad, Bettina Helfrich, Jeffrey Lagarias, and Claus P. Schnorr. “Polynomial Time Algorithms for Finding Integer Relations Among Real Numbers”. *STACS 86*. Ed. by B. Monien and G. Vidal-Naquet. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 105–118 (cit. on p. 110).
- [Hazay, 2010] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Springer Science & Business Media, 2010 (cit. on p. 173).
- [He, 2017] Yue He, Fen Cheng, Dai-Wen Pang, and Hong-Wu Tang. “Colorimetric and Visual Determination of DNase I Activity Using Gold Nanoparticles as an Indicator”. *Microchimica Acta* 184.1 (2017), pp. 101–106 (cit. on p. 67).
- [Heap, 2015] Winston Heap and Sofia Lindqvist. *Moments of Random Multiplicative Functions and Truncated Characteristic Polynomials*. 2015. arXiv: 1505.03378 [math.NT] (cit. on pp. 14, 99–101).
- [Higgins, 1989] Gerald L. Higgins. “The History of Confidentiality in Medicine”. *Canadian Family Physician* 35 (1989), p. 921 (cit. on p. 3).
- [Hogan, 2020] Catherine A. Hogan, Malaya K. Sahoo, and Benjamin A. Pinsky. “Sample Pooling as a Strategy to Detect Community Transmission of SARS-COV-2”. *Jama* 323.19 (2020), pp. 1967–1969 (cit. on p. 22).
- [Hollenstein, 2020] Marcel Hollenstein, David Naccache, Peter B. Rønne, Peter Y. A. Ryan, Robert Weil, and Ofer Yifrach-Stav. “Preservation of DNA Privacy During the Large Scale Detection of Covid-19”. *arXiv preprint arXiv:2007.09085* (2020).
- [Homer, 2008] Nils Homer, Szabolcs Szeling, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V. Pearson, Dietrich A Stephan, Stanley F. Nelson, and David W. Craig. “Resolving Individuals Contributing Trace Amounts of DNA to Highly Complex Mixtures using High-Density SNP Genotyping Microarrays”. *PLoS genetics* 4.8 (2008) (cit. on p. 65).

- [Horster, 1994] Patrick Horster, Holger Petersen, and Markus Michels. “Meta-ElGamal Signature Schemes”. *Proceedings of the 2nd ACM Conference on Computer and Communications Security*. CCS ’94. Fairfax, Virginia, USA: Association for Computing Machinery, 1994, pp. 96–107 (cit. on p. 193).
- [Huang, 1996] Zeqi Huang, Michael J. Fasco, and Laurence S. Kaminsky. “Optimization of DNase I Removal of Contaminating DNA from RNA for Use in Quantitative RNA-PCR”. *Biotechniques* 20.6 (1996), pp. 1012–1020 (cit. on p. 67).
- [Hwang, 1972] FK Hwang. “A Method for Detecting all Defective Members in a Population by Group Testing”. *Journal of the American Statistical Association* 67.339 (1972), pp. 605–608 (cit. on p. 20).
- [India-Today, 2020] India-Today. *Centre Allows Covid-19 Pool Testing, Plasma Therapy in Maharashtra*. 2020 (cit. on p. 22).
- [Jainsky, 2023] Julien S. Jainsky, David Naccache, Bassem Ouni, and Ofer Yifrach-Stav. “Authenticating Medications with QR-Codes and Compact Digital Signatures”. *IACR Cryptol. ePrint Arch.* (2023), p. 1817.
- [Kang, 2011] Min-Jung Kang, Hannah Yu, Sook-Kyung Kim, Sang-Ryoul Park, and Inchul Yang. “Quantification of Trace-Level DNA by Real-Time Whole Genome Amplification”. *PLoS one* 6.12 (2011) (cit. on p. 61).
- [Kaur, 2021] Harpreet Kaur, Xuejun Liu, Prabin Nepal, Subakar Paramanatham, Monika Garg, Zemin Li, Andy Tjeng, Moises Rodriguez, Laura Jong, Azam Mizrahoossein, Ofer Yifrach-Stav, Erin Bassett, Randy Ringuette, Charles Campbell, and Melanie Kelly. *Methods, Processes, and Compositions for Improved Preparation of hu308 and hu433*. Publication number: WO2022061461A1, filed: Sep, 2020, published: Sep, 2021.
- [Kayser, 2011] Manfred Kayser and Peter De Knijff. “Improving Human Forensics Through Advances in Genetics, Genomics and Molecular Biology”. *Nature Reviews Genetics* 12.3 (2011), p. 179 (cit. on p. 61).
- [Kim, 2005] Ki Hang Kim and Fred W. Roush. *Factorization of polynomials in one variable over the tropical semiring*. 2005 (cit. on p. 189).
- [Labelle, 1990] Jacques Labelle and Yeong-Nan Yeh. “Generalized dyck paths”. *Discrete mathematics* 82.1 (1990), pp. 1–6 (cit. on p. 10).
- [Lee, 2000] Bonnie M. Lee. *Comparison of FDA and HHS Human Subject Protection Regulations*. 2000 (cit. on p. 62).
- [Lee, 2020] Ethan Lee and Virginia L. Ma. “At Least 500,000 Tests Needed Per Day to Reopen Economy, Harvard Researchers Say”. *The Harvard Crimson* (2020) (cit. on p. 19).
- [Lenstra, 1982] Arjen K. Lenstra, Hendrik W. Lenstra Jr., and László Lovász. “Factoring Polynomials with Rational Coefficients”. *Mathematische Annalen* 261.4 (1982), pp. 515–534 (cit. on pp. 84, 110).
- [Li, 1962] Chou Hsiung Li. “A Sequential Method for Screening Experimental Variables”. *Journal of the American Statistical Association* 57.298 (1962), pp. 455–477 (cit. on p. 20).
- [Litvak, 1994] Eugene Litvak, Xin M. Tu, and Marcello Pagano. “Screening for the Presence of a Disease by Pooling Sera Samples”. *Journal of the American Statistical Association* 89.426 (1994), pp. 424–434 (cit. on p. 20).
- [Liu, 2020] Jia Liu, Yi Chen, Kefan Xie, and Xiaohong Chen. “Is Pool Testing Method of Covid-19 Employed in Germany and India Effective?” (2020) (cit. on p. 22).
- [MRaïhi, 1996] David M’Raïhi and David Naccache. “Batch Exponentiation: A Fast DLP-Based Signature Generation Strategy”. *Proceedings of the 3rd ACM Conference on Computer and Communications Security*. CCS ’96. New Delhi, India: Association for Computing Machinery, 1996, pp. 58–61 (cit. on p. 186).
- [Malin, 2000] Bradley Malin and Latanya Sweeney. “Determining the Identifiability of DNA Database Entries.” *Proceedings of the AMLA Symposium*. American Medical Informatics Association. 2000, p. 537 (cit. on p. 62).
- [Malin, 2001] Bradley Malin and Latanya Sweeney. “Re-Identification of DNA Through an Automated Linkage Process.” *Proceedings of the AMLA Symposium*. American Medical Informatics Association. 2001, p. 423 (cit. on p. 62).
- [Master, 2012] Zubin Master, Erin Nelson, Blake Murdoch, and Timothy Caulfield. “Biobanks, Consent and Claims of Consensus”. *Nature Methods* 9.9 (2012), p. 885 (cit. on p. 61).
- [McMahan, 2012] Christopher S. McMahan, Joshua M. Tebbs, and Christopher R. Bilder. “Informative Dorfman Screening”. *Biometrics* 68.1 (2012), pp. 287–296 (cit. on p. 20).
- [Merkle, 1987] Ralph C Merkle. “Protocols for Public Key Cryptosystems”. *Proceedings of the IEEE* 75.1 (1987), pp. 56–62 (cit. on p. 175).
- [Micali, 2006] Silvio Micali and Rafael Pass. “Local Zero Knowledge”. *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*. 2006, pp. 306–315 (cit. on p. 174).
- [Micali, 1990] Silvio Micali and Adi Shamir. “An Improvement of the Fiat-Shamir Identification and Signature Scheme”. *Advances in Cryptology — CRYPTO’88*. Ed. by Shafi Goldwasser. New York, NY: Springer New York, 1990, pp. 244–247 (cit. on pp. 18, 173).
- [Mohamed, 2016] Mohamed Saied Emam Mohamed and Albrecht Petzoldt. *The Shortest Signatures Ever*. Cryptology ePrint Archive, Paper 2016/911. 2016 (cit. on p. 186).

- [Monica, 2017] Kate Monica. “79K Patients Affected by Emory Healthcare Data Breach.” *TechTarget, Latest Health Data Breaches News* (2017) (cit. on p. 62).
- [Morris, 2006] Max D. Morris. “An Overview of Group Factor Screening”. *Screening*. Springer, 2006, pp. 191–206 (cit. on p. 20).
- [Mosher, 2019a] Steven W. Mosher. “China’s Ploy to Establish a Global DNA Database - The First in a 3-Part Series on the Regime’s DNA Collection Project”. *The Epoch Times* (2019) (cit. on p. 62).
- [Mosher, 2019b] Steven W. Mosher. “What Will China Do With Your DNA? China’s Fourth Magic Weapon, Part III: Bioweapons”. *The Epoch Times* (2019) (cit. on p. 62).
- [Muanalifah, 2021] Any Muanalifah and Sergei Sergeev. “On the tropical discrete logarithm problem and security of a protocol based on tropical semidirect product”. *Communications in Algebra* 50.2 (2021), pp. 861–879 (cit. on p. 189).
- [Naccache, 1992] David Naccache and Patrice Frémanteau. “Unforgeable Identification Device, Identification Device Reader and Method of Identification”. Patent EP19930112678. 1992 (cit. on p. 181).
- [Naccache, 2001] David Naccache and Jacques Stern. “Signing on a Postcard”. *Financial Cryptography*. Ed. by Yair Frankel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, 121–135 (cit. on p. 186).
- [Naccache, 2022a] David Naccache and Ofer Yifrach-Stav. “A Conjecture From a Failed Cryptanalysis”. *IACR Cryptol. ePrint Arch.* (2022), p. 1273.
- [Naccache, 2022b] David Naccache and Ofer Yifrach-Stav. “Invisible Formula Attacks”. *IACR Cryptol. ePrint Arch.* (2022), p. 1110.
- [Naccache, 2022c] David Naccache and Ofer Yifrach-Stav. “Invisible Formula Attacks”. The Black Hat (Las Vegas, USA). 2022.
- [Naccache, 2022d] David Naccache and Ofer Yifrach-Stav. “On Squaring Modulo Mersenne Numbers”. *IACR Cryptol. ePrint Arch.* (2022), p. 1197.
- [Naccache, 2022e] David Naccache and Ofer Yifrach-Stav. “Pattern Recognition Experiments on Mathematical Expressions”. *arXiv preprint arXiv:2301.01624* (2022).
- [Naccache, 2023a] David Naccache and Ofer Yifrach-Stav. “A Note on Moments of Random Multiplicative Functions and Truncated Characteristic Polynomials”. *HAL 04089572* (2023).
- [Naccache, 2023b] David Naccache and Ofer Yifrach-Stav. “Continued Fractions With Numerator $(v + 1 + n)(v + 2 + n)(v + 3 + n)(v + 4 + n)$ ”. *HAL 04092469* (2023).
- [Naccache, 2023c] David Naccache and Ofer Yifrach-Stav. “Explicit Formulae for Some Polynomial Continued Fractions”. *HAL 04105623* (2023).
- [Naccache, 2023d] David Naccache and Ofer Yifrach-Stav. “On Catalan Constant Continued Fractions”. *Codes, Cryptology and Information Security*. Ed. by Said El Hajji, Sihem Mesnager, and El Mamoun Souidi. Cham: Springer Nature Switzerland, 2023, pp. 43–54 (cit. on p. 105).
- [Naccache, 2023e] David Naccache and Ofer Yifrach-Stav. “On Catalan Constant Continued Fractions”. *Codes, Cryptology and Information Security*. Ed. by Saïd El Hajji, Sihem Mesnager, and El Mamoun Souidi. Cham: Springer Nature Switzerland, 2023, pp. 43–54.
- [Naccache, 2023f] David Naccache and Ofer Yifrach-Stav. “On The Practical Advantage of Committing Challenges in Zero-Knowledge Protocols”. *Cryptology ePrint Archive, Paper 2023/1961* (2023).
- [Naccache, 2023g] David Naccache and Ofer Yifrach-Stav. “The Balkans Continued Fraction”. *arXiv preprint arXiv: 2308.06291* (2023).
- [Neven, 2009] Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. “Hash Function Requirements for Schnorr Signatures”. *Journal of Mathematical Cryptology* 3.1 (2009), pp. 69–87 (cit. on p. 187).
- [Nevo, 2017] Eshkol Nevo. *Three Floors Up*. New York: Other Press, 2017 (cit. on p. 3).
- [Nguyen, 2019] Ngoc T. Nguyen, Hrayr Aprahamian, Ebru K. Bish, and Douglas R. Bish. “A Methodology for Deriving the Sensitivity of Pooled Testing, Based on Viral Load Progression and Pooling Dilution”. *Journal of translational medicine* 17.1 (2019), p. 252 (cit. on pp. 2, 21).
- [Nimbran, 2018] Amrik Singh Nimbran and Paul Levrie. *Some Continued Fractions for π and G* . 2018. arXiv: 1806.03346 [math.HO] (cit. on p. 120).
- [Norris, 2017] Gareth Norris. “Authoritarianism and Privacy: The Moderating Role of Terrorism”. *Surveillance & Society* 15.3/4 (2017), pp. 573–581 (cit. on p. 4).
- [Ouafi, 2009] Khaled Ouafi and Serge Vaudenay. “Smashing SQUASH-0”. *Advances in Cryptology - EUROCRYPT 2009*. Ed. by Antoine Joux. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 300–312 (cit. on p. 172).
- [Overstreet, 2019] Kim Overstreet. *\$200 Billion Pharma Counterfeit Drug Market Growing by 20% Per Year*. 2019. URL: <https://www.healthcarepackaging.com/news/traceability-serialization/article/21102806/200-billion-pharma-counterfeit-drug-market-growing-by-20-per-year> (cit. on p. 181).

- [Panny, 2023] Lorenz Panny. *Forging tropical signatures*. Cryptology ePrint Archive, Paper 2023/1748. 2023 (cit. on p. 189).
- [Pass, 2015] Rafael Pass. “A Tutorial on Concurrent Zero-Knowledge”. *IACR Cryptology ePrint Archive* 2015 (2015), p. 543 (cit. on p. 176).
- [Pasternack, 1974] Bernard Pasternack, Milton Sobel, and James Thomas. *Group-testing, Halving Procedures and Binary Search*. Tech. rep. University of Minnesota, 1974 (cit. on p. 2).
- [Pecci, 2017] Alexandra Wilson Pecci. “Healthcare Data Breaches Up 40% Since 2015”. *HealthLeaders* (2017) (cit. on p. 62).
- [Pornin, 2022] Thomas Pornin. *Truncated EdDSA/ECDSA Signatures*. Cryptology ePrint Archive, Paper 2022/938. 2022 (cit. on p. 186).
- [Press, 2019] Rich Press. “DNA Mixtures: A Forensic Science Explainer”. *NIST US Government Publications* (2019) (cit. on p. 64).
- [Raayoni, 2021] Gal Raayoni, Shahar Gottlieb, Yahel Manor, George Pisha, Yoav Harris, Uri Mendlovic, Doron Haviv, Yaron Hadad, and Ido Kaminer. “Generating Conjectures on Fundamental Constants with the Ramanujan Machine”. *Nature* 590.7844 (2021), pp. 67–73 (cit. on pp. 5, 71, 84, 106).
- [Raayoni, 2019] Gal Raayoni, George Pisha, Yahel Manor, Uri Mendlovic, Doron Haviv, Yaron Hadad, and Ido Kaminer. “The Ramanujan Machine: Automatically Generated Conjectures on Fundamental Constants”. *CoRR* abs/1907.00205 (2019). arXiv: 1907.00205 (cit. on pp. 71, 84).
- [Rivest, 1978] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. *Commun. ACM* 21.2 (1978), pp. 120–126 (cit. on p. 163).
- [Robinson, 2021] Karen R Robinson. “Comparing the Spanish Flu and Covid-19 Pandemics: Lessons to Carry Forward”. *Nursing Forum*. Vol. 56. 2. Wiley Online Library. 2021, pp. 350–357 (cit. on p. 1).
- [Roth, 1999] Willi Kurt Roth, Marijke Weber, and Erhard Seifried. “Feasibility and Efficacy of Routine PCR Screening of Blood Donations for Hepatitis C Virus, Hepatitis B Virus, and HIV-1 in a Blood-Bank Setting”. *The Lancet* 353.9150 (1999), pp. 359–363 (cit. on pp. 2, 22).
- [Rowe, 2006] Rosemary Rowe and Michael Calnan. “Trust Relations in Health Care — the New Agenda”. *The European Journal of Public Health* 16.1 (2006), pp. 4–6 (cit. on p. 68).
- [Ryan, 2020] Matt Ryan. “Gov. Ricketts Provides Update on Coronavirus Testing”. *3KMTV* (2020) (cit. on p. 22).
- [Salecl, 2002] Renata Salecl. “The Exposure of Privacy in Today’s Culture”. *Social Research: An International Quarterly* 69.1 (2002), pp. 1–8 (cit. on p. 3).
- [Sanderson, 1979] Danny Sanderson. טוֹב עִם סְאוֹנֵר וְדָנִי עִם אֵמָא וְדָנִי www.youtube.com/watch?v=0nrKa4H7Foc. 1979 (cit. on p. 134).
- [Sato, 2014] Shinobu Sato and Shigeori Takenaka. “Highly Sensitive Nuclease Assays Based on Chemically Modified DNA or RNA”. *Sensors* 14.7 (2014), pp. 12437–12450 (cit. on p. 67).
- [Schulman, 2020] Julia Schulman. “Israel Renegotiates Covid-19 Testing Lab Deal With China”. *Foundation for Defense of Democracies* (2020) (cit. on p. 62).
- [Schwartz, 2020] Matthew S. Schwartz. “Germany Backs Away from Compiling Coronavirus Contacts in a Central Database”. *NPR* (2020) (cit. on p. 61).
- [Shamir, 2008] Adi Shamir. “SQUASH – A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags”. *Fast Software Encryption*. Ed. by Kaisa Nyberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 144–157 (cit. on p. 171).
- [Shani-Narkiss, 2020] Haran Shani-Narkiss, Omri David Gilday, Nadav Yaron, and Itamar Daniel Landau. “Efficient and Practical Sample Pooling High-Throughput PCR Diagnosis of Covid-19”. *medRxiv* (2020) (cit. on p. 22).
- [Simmons, 1984] Gustavus J. Simmons. “A System for Verifying User Identity and Authorization at the Point-of-Sale or Access”. *Cryptologia* 8.1 (1984), pp. 1–21 (cit. on p. 181).
- [Simmons, 1991] Gustavus J. Simmons. “Identification of Data, Devices, Documents and Individuals”. *Proceedings. 25th Annual 1991 IEEE International Carnahan Conference on Security Technology*. 1991, pp. 197–218 (cit. on p. 181).
- [Sinnott-Armstrong, 2020] Nasa Sinnott-Armstrong, Daniel Klein, and Brendan Hickey. “Evaluation of Group Testing for SARS-CoV-2 RNA”. *medRxiv* (2020) (cit. on pp. 22, 30).
- [Sloane, 2023] NJA Sloane. “A Handbook of Integer Sequences” Fifty Years Later”. *The Mathematical Intelligencer* (2023), pp. 1–13 (cit. on p. 11).
- [Sobel, 1959] Milton Sobel and Phyllis A. Groll. “Group Testing to Eliminate Efficiently all Defectives in a Binomial Sample”. *Bell System Technical Journal* 38.5 (1959), pp. 1179–1252 (cit. on pp. 20, 21).

- [Sterrett, 1957] Andrew Sterrett. “On the Detection of Defective Members of Large Populations”. *The Annals of Mathematical Statistics* 28.4 (1957), pp. 1033–1036 (cit. on p. 20).
- [Su, 2013] Xin Su, Chen Zhang, Xiaocui Zhu, Simin Fang, Rui Weng, Xianjin Xiao, and Meiping Zhao. “Simultaneous Fluorescence Imaging of the Activities of DNases and 3’ Exonucleases in Living Cells with Chimeric Oligonucleotide Probes”. *Analytical chemistry* 85.20 (2013), pp. 9939–9946 (cit. on p. 67).
- [Täufer, 2020] Matthias Täufer. “Rapid, Large-Scale, and Effective Detection of Covid-19 via Non-Adaptive Testing”. *bioRxiv* (2020). eprint: <https://www.biorxiv.org/content/early/2020/04/13/2020.04.06.028431.full.pdf> (cit. on p. 22).
- [Taylor, 2014] Duncan Taylor, Jo-Anne Bright, and John Buckleton. “Interpreting Forensic DNA Profiling Evidence Without Specifying the Number of Contributors”. *Forensic Science International: Genetics* 13 (2014), pp. 269–280 (cit. on p. 66).
- [Taylor, 2010] Steve M. Taylor, Jonathan J. Juliano, Paul A. Trottman, Jennifer B. Griffin, Sarah H. Landis, Paluku Kitsa, Antoinette K. Tshetu, and Steven R. Meshnick. “High-Throughput Pooling and Real-Time PCR-Based Strategy for Malaria Detection”. *Journal of clinical microbiology* 48.2 (2010), pp. 512–519 (cit. on p. 22).
- [Tomkins, 2007] Oren Tomkins, Ofer Friedman, Sebastian Ivens, Clemens Reiffurth, Sebastian Major, Jens Peter Dreier, Uwe Heineemann, and Alon Friedman. “Blood–Brain Barrier Disruption Results in Delayed Functional and structural Alterations in the Rat Neocortex”. *Neurobiology of disease* 25.2 (2007), pp. 367–377.
- [Torres, 2020] Ignacio Torres, Eliseo Albert, and David Navarro. “Pooling of Nasopharyngeal Swab Specimens for SARS-COV-2 Detection by RT-PCR”. *Journal of Medical Virology* (2020) (cit. on p. 22).
- [Ungar, 1960] Peter Ungar. “The Cutoff Point for Group Testing”. *Comm. Pure Appl. Math.* 13.1 (1960), pp. 49–54 (cit. on p. 26).
- [Van, 2012] Tam T. Van, Joseph Miller, David M. Warshauer, Erik Reisdorf, Daniel Jernigan, Rosemary Humes, and Peter A. Shult. “Pooling Nasopharyngeal/Throat Swab Specimens to Increase Testing Capacity for Influenza Viruses by PCR”. *Journal of clinical microbiology* 50.3 (2012), pp. 891–896 (cit. on p. 2, 22).
- [Vaudenay, 2020] Serge Vaudenay. “Centralized or Decentralized”. *The contact tracing dilemma* (2020) (cit. on p. 61).
- [Von Mises, 1939] Richard Von Mises. *Über aufteilungs-und besetzungswahrscheinlichkeiten*. na, 1939 (cit. on p. 8).
- [Wagner, 2002] David Wagner. “A Generalized Birthday Problem”. *Annual International Cryptology Conference*. Springer. 2002, pp. 288–304 (cit. on p. 9).
- [Waldo, 2010] Ann Waldo. “The Texas Newborn Bloodspot Saga has Reached a Sad—and Preventable—Conclusion”. *Genomics Law Report* 16 (2010), pp. 1–45 (cit. on p. 61).
- [Wang, 2020] Yishan Wang, Hanyujie Kang, Xuefeng Liu, and Zhaohui Tong. “Combination of RT-qPCR Testing and Clinical Features for Diagnosis of Covid-19 Facilitates Management of SARS-COV-2 Outbreak”. *Journal of Medical Virology* (2020) (cit. on p. 19).
- [Westin, 1967] Alan F. Westin. “Special Report: Legal Safeguards to Insure Privacy in a Computer Society”. *Communications of the ACM* 10.9 (1967), pp. 533–537 (cit. on p. 3).
- [Westin, 2003] Alan F. Westin. “Social and Political Dimensions of Privacy”. *Journal of social issues* 59.2 (2003), pp. 431–453 (cit. on pp. 3, 4).
- [Wiame, 2000] Ilse Wiame, Serge Remy, Rony Swennen, and László Sági. “Irreversible Heat Inactivation of DNase I without RNA Degradation”. *Biotechniques* 29.2 (2000), pp. 252–256 (cit. on p. 67).
- [Williams, 2018] Jane Williams. *Exploring The World’s Largest Biobanks*. 2018 (cit. on p. 61).
- [Woolf, 2020] Peter Woolf. *Origami Essays*. 2020 (cit. on p. 22).
- [World Health Organization, 2017] World Health Organization. “A Study on the Public Health and Socioeconomic Impact of Substandard and Falsified Medical Products” (2017) (cit. on p. 181).
- [Xu, 2007] Xiaoyang Xu, Min Su Han, and Chad A. Mirkin. “A Gold-Nanoparticle-Based Real-Time Colorimetric Screening Method for Endonuclease Activity and Inhibition”. *Angewandte Chemie International Edition* 46.19 (2007), pp. 3468–3470 (cit. on p. 67).
- [Yelin, 2020] Idan Yelin, Noga Aharony, Einat Shaer-Tamar, Amir Argoetti, Esther Messer, Dina Berenbaum, Einat Shafran, Areen Kuzli, Nagam Gandali, Tamar Hashimshony, Yael Mandel-Gutfreund, Michael Halberthal, Yuval Geffen, Moran Szwarcwort-Cohen, and Roy Kishony. “Evaluation of Covid-19 RT-qPCR Test in Multi-Sample Pools”. *medRxiv* (2020) (cit. on p. 22).
- [Yifrach-Stav, 2023] Ofer Yifrach-Stav. “Can medical devices be used to rob banks?” *IPA - International Pharmaceutical Academy* (2023).
- [Young, 1996] Adam Young and Moti Yung. “The Dark Side of “Black-Box” Cryptography or: Should We Trust Capstone?” *Advances in Cryptology — CRYPTO ’96*. Ed. by Neal Koblitz. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 89–103 (cit. on p. 169).
- [Zaaijer, 2017] Sophie Zaaijer, Assaf Gordon, Daniel Speyer, Robert Piccone, Simon Cornelis Groen, and Yaniv Erlich. “Rapid Re-Identification of Human Samples Using Portable DNA Sequencing”. *Elife* 6 (2017), e27798 (cit. on p. 61).

[Zhang, 2022] Shengtong Zhang. *On a Conjecture From a Failed Cryptanalysis*. Cryptology ePrint Archive, Paper 2022/1287. 2022 (cit. on pp. 5, 103).

RÉSUMÉ

Cette thèse est le fruit de recherches menées entre 2019 et 2023, réparties en trois parties. Dans la première partie, nous étudions des questions liées à la pandémie du Covid-19, telles que les tests par lots, un domaine bien établi dans lequel les échantillons de plusieurs patients sont mélangés pour effectuer des tests collectifs. Une telle procédure permet de réduire les coûts et d'économiser du temps. Nous proposons des algorithmes tenant compte des probabilités *a priori* que les tests individuels soient positifs. De telles probabilités peuvent être évaluées lors d'un examen clinique préalable du patient. Nous examinons également les tests par lots en situation d'urgence, où certains échantillons doivent être analysés en priorité. Dans les deux cas, nous proposons de nouveaux algorithmes et les analysons en détail. Cette section traite également de la préservation de la confidentialité de l'ADN dans les tests de dépistage du Covid-19. Dans la deuxième partie, nous présentons nos résultats en mathématiques expérimentales, où nous avons découvert plusieurs nouvelles conjectures sur les fractions continues grâce à des explorations automatisées. Toutes ces conjectures ont été testées numériquement pour évaluer leur plausibilité. Enfin, dans la troisième partie de la thèse, nous abordons divers résultats dans le domaine de la sécurité informatique, tels qu'une attaque inconnue jusqu'à présent sur le logiciel Mathematica, un nouveau mécanisme de protection contre les médicaments contrefaits, et des nouvelles observations sur les preuves à divulgation nulle.

MOTS CLÉS

Sécurité de l'information, Tests par lots, Mathématiques expérimentales, Confidentialité de l'ADN, Covid-19

ABSTRACT

This thesis is the culmination of research conducted between 2019 and 2023. It is divided into three parts. In the first part, we explore algorithms related to the Covid-19 pandemic, such as Pool Testing, a well-established technique where samples from multiple patients are pooled for collective testing, allowing for cost reduction and time savings. We propose algorithms taking into account the *a priori* probabilities that individual tests are positive, which can be evaluated during a prior clinical examination of the patient. We also examine Pool Testing in emergency situations, where certain samples need to be analyzed according to some prescribed priority order. In both cases, we propose new algorithms and analyze them in detail. This section also deals with DNA privacy preservation in Covid-19 tests. In the second part, we present our results in experimental mathematics, where we have discovered several new conjectures on continued fractions through automated exploration. All those conjectures have been numerically tested to assess their plausibility. Finally, the third part of this thesis is devoted to various results in the field of computer security, such as a previously unknown attack on the Mathematica software, a new protection mechanism against counterfeit medication, and new observations on zero-knowledge proofs.

KEYWORDS

Information Security, Pool Testing, Experimental Mathematics, DNA Privacy, Covid-19