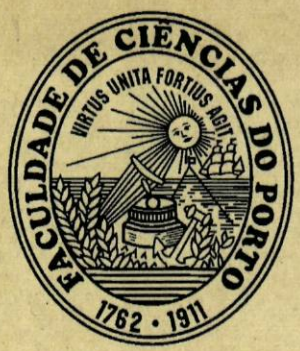
Alexandre Jorge Teixeira Miranda Pinto

Applications of Kolmogorov Complexity to Cryptography



Departamento de Ciências de Computadores
Faculdade de Ciências da Universidade do Porto
2007

Alexandre Jorge Teixeira Miranda Pinto

Applications of Kolmogorov Complexity to Cryptography

Tese submetida à Faculdade de Ciências da
Universidade do Porto para obtenção do grau de Doutor
em Ciência de Computadores

Departamento de Ciências de Computadores
Faculdade de Ciências da Universidade do Porto
2007

Acknowledgments

I am indebted to many people who have supported me during these four years. First of all, I want to thank my advisors Armando Matos and Luís Antunes who taught me all the basics abouts research. They introduced me to the fields of Kolmogorov and Computational Complexity and guided me in my first steps in these areas. I thank them for their patience in answering my questions when all still seemed so strange and vague. They tirelessly reviewed my work, pointing my mistakes and generally showing me the best way to improve, and allowed me to develop and nurture my love for Cryptography. They taught me all I know about writing papers and having fun with science at the same time.

I wish to thank all my coauthors during these years, Luís Antunes, Lance Fortnow, Sophie Laplante, Armando Matos, Liliana Salvador, André Souto, for letting me use our papers for my thesis.

I also want to thank Lance Fortnow for receiving me in Chicago and showing me how much I had yet to learn. I took many valuable lessons from conversations with him and from his clever proofs and ideas.

I am also enormously grateful to Sophie Laplante for letting me stay at the LRI for almost a month. It was extremely pleasant to work with her and I owe Sophie some important ideas for this thesis. A word of gratitude goes to all the people I met there and with whom I had so many interesting conversations: Troy, Marc, Vincent, Loïc, Frédéric and all the others whose names I sadly have not kept but who made this one of the best times in my PhD. My thanks also go to Woo-Jun Park for being such an excellent office companion while I was there.

I thank Jonathan Katz and Moses Liskov for crucial insights; Manuel Barbosa and Pooya Farshim for many ideas and discussions regarding cryptography in general and all the people I met at DCC during these years, in particular, Jorge, David, Hugo, Nuno, Marco, André and Andreia. I also extend my thanks to Patrick Plouffe for revising the French version of the abstract.

For financial support, I thank the Fundação para a Ciência e Techologia, LIACC and the Fundação Luso-Americana para o Desenvolvimento.

I would not have survived this PhD if it had not been for the world outside resarch. I have a bottomless debt of gratitude to my friends Plácido, César, Zé Maria, Ana Maria, Nuno, Hélder and Ana. Our endless adventures in the worlds of D&D always

served as a sanity check and an escape from the sometimes hard life of research, an outlet for fantasy and free creative thought. I cherish all the hours I spent devising the next twist in their adventures in the Land of Mists or dealing with the moral dilemmas of my dear Knight of Solamnia. I also want to thank Filipe, Hugo Silva and Hugo Ferreira for all the good times and the promise of future adventures and my teacher Carlos Teixeira for having such a high confidence in me, even more than myself. His support was of the utmost importance for my self-esteem.

I leave for the end those to whom I owe the most, those that were always indefatigably supporting me through the hardships. My sister Ana and my parents were the ones who never doubted I could do it and gave me the strength to continue at those times when all seemed lost and useless. Through the years, they taught me two very important lessons: to listen others and value their opinion, but to analyse anything they say before accepting it.

I can not express all the support of my wife Linda who listened to all my doubts and conflicts. She was always understanding, a source of companionship and renewed strength and more than anyone else the person that made me persevere and fight for success until the end. I owe much of this PhD to her patience, trust and love.

Abstract

The corner stone of modern cryptography is the notion of proof of security. There are two basic notions of security: information-theoretic security and computational security. Schemes which have the first kind of security are unconditionally secure: no matter how powerful the adversary is, the scheme will be secure. Schemes with computational security are dependent on the validity of some computational assumption, stating that a certain problem is hard for a resource-bounded adversary (the bound is usually time).

Proofs of security usually try to establish that a given scheme is secure against a specific kind of attack by stating that the advantage the attacker gains by knowing some specific information is negligible when measured over all instances of the scheme. This advantage is measured as the difference between the probabilities of success that the adversary has in two situations where a single factor changes. For example, it could be the difference between the probability of guessing a plaintext when she knows the ciphertext or not; or it could be the difference between the probabilities of choosing each one between two possible plaintext messages for a given ciphertext.

In this thesis, we introduce the evaluation of security of a cryptographic scheme at the level of its instances by defining a different notion of advantage. Instead of considering the difference of two probabilities, we define it as the difference between the amounts of information necessary to break a particular instance when some extra knowledge is known, where information is defined according to the theory of Kolmogorov complexity. The security of the system can then be measured by the average security of its instances, being that the system is secure if the expected value of the information gained by the extra knowledge is low.

Our results are divided in three parts. In the first one, we analyse three systems that have unconditional security. For these, we define the notion of individual security and show that, if sufficiently many instances are individually secure, then the average

information-theoretic advantage is also low. This suggests that the individual notion is more refined than the traditional global one.

In the second part, we show that information-theoretic commitment systems can be built from a composition of a secure cypher and a secure authentication system. We also show that optimal commitment systems can also be decomposed in a pair of unconditionally secure cypher and authentication systems, thus showing that this composition is the only way of building optimal commitment systems.

In the third part, we address the possibility of extending the analysis done in the first part to systems that have only computational proofs of security, by comparing different notions of computational entropy with the expected value of time bounded Kolmogorov complexity. The aim is to find a parallel to the relation between Shannon's entropy and the expected value of unbounded Kolmogorov complexity. We show that at least half of this relation exists.

Resumo

A base da criptografia moderna é a noção de prova de segurança. Há dois tipos básicos de segurança: segurança baseada na teoria da informação e segurança computacional. Esquemas com o primeiro tipo de segurança são incondicionalmente seguros: o esquema é seguro independentemente do poder computacional do adversário. Esquemas com segurança computacional dependem da validade de uma assunção computacional, afirmando que um dado problema é difícil para um adversário com recursos limitados (em que usualmente o recurso é o tempo).

Normalmente, as provas de segurança tentam estabelecer que um dado esquema é seguro contra um ataque específico ao garantir que a vantagem que um atacante ganha por saber uma dada informação é negligenciável quando medida sobre todas as instâncias do esquema. Esta vantagem é medida como a diferença entre as probabilidades de sucesso que o adversário tem em duas situações onde um único factor muda. Por exemplo, pode ser a diferença entre a probabilidade quando o atacante conhece o texto cifrado e aquela quando este é desconhecido; ou pode ser a diferença entre as probabilidades de escolher cada um de dois possíveis textos originais para um dado texto cifrado.

Nesta tese, apresentamos a avaliação de segurança de um esquema criptográfico ao nível das suas instâncias, definindo uma noção de vantagem diferente. Em vez de considerarmos a diferença entre duas probabilidades, definimos vantagem como sendo a diferença entre a quantidade de informação necessária para quebrar uma dada instância nas duas situações diferenciadas pelo conhecimento adicional do atacante, onde a informação é definida de acordo com a teoria da complexidade de Kolmogorov. A segurança do sistema pode então ser medida pela média da segurança das suas instâncias, sendo que o sistema é seguro se o valor esperado da informação obtida com o conhecimento da informação extra for baixo.

Os resultados dividem-se em três partes. Na primeira, analisamos três sistemas que

têm segurança incondicional. Para estes, definimos uma noção de segurança individual e mostramos que, se um número suficiente de instâncias forem seguras, então a vantagem média de acordo com a teoria da informação é próxima de 0. Isto sugere que a noção individual é mais refinada que a versão global tradicional.

Na segunda parte, mostra-se que sistemas de compromisso seguros segundo a teoria da informação podem ser construídos a partir de uma composição de um sistema de cifra e um sistema de autenticação seguros. Também se mostra que qualquer sistema de compromisso óptimo pode ser decomposto num par de um sistema de cifra e um sistema de autenticação seguros, mostrado assim que esta composição é a única forma de construir sistemas de compromisso óptimos.

Na terceira parte, estudamos a possibilidade de generalizar a análise feita na primeira parte a sistemas que apenas têm provas de segurança computacional, comparando diferentes noções de entropia computacional com o valor esperado da complexidade de Kolmogorov limitada no tempo. O objectivo é encontrar uma relação paralela à existente entre a entropia de Shannon e o valor esperado da complexidade de Kolmogorov sem limites de tempo. Mostramos que pelo menos metade desta relação é válida.

Résumé

La cryptographie moderne est basée sur la notion de preuve de la sécurité. Il y a deux types basiques de sécurité: la sécurité en accord avec la théorie de l'information ainsi que la sécurité computationelle. Les schémas du premier type de sécurité sont inconditionnellement sécuritaires, même si l'adversaire est computationellement tout-puissant. Par contre, la sécurité des schémas avec sécurité computationelle dépend de la validité d'une supposition computationelle affirmant qu'un certain problème est difficile pour un adversaire avec des ressources limités, où la ressource utilisée est, en général, le temps.

D'habitude, les preuves de la sécurité essaient d'établir qu'un certain schéma est sécuritaire contre une attaque spécifique en garantissant que l'avantage que l'attaquant gagne, parce qu'il connaît une certaine information, est négligeable quand elle est mesurée sur toutes les instances possibles du schéma. Cet avantage est la différence entre les probabilités de succès de l'adversaire en deux situations distinctes, où un seul facteur est changé. Par exemple, cela pouvait être la différence entre les probabilités de trouver le texte clair quand le texte chiffré est connu et quand il ne l'est pas; ou cela pouvait être la différence entre la probabilité de choisir chaqu'un d'entre deux possibles textes clairs pour un certain texte chifré.

Dans cette thèse, nous présentons une évaluation de sécurité d'un schéma cryptographique au niveau de ses instances, définissant une notion d'avantage différente. Elle n'est plus la différence entre deux probabilités, mais la différence entre la quantité d'information nécessaire pour briser une certaine instance dans les deux situations distinguées par une connaissance additionelle de l'adversaire, où cette information est définie en accord avec la théorie de la complexité de Kolmogorov. La sécurité du système peut alors être définie comme la moyenne de la sécurité de ses instances, et le système sera sécuritaire si la valeur attendue de l'information obtenue avec la connaissance additionelle est petite.

Nos résultats sont divisés en trois parties. Dans la première, on analyse trois systèmes possédant de la sécurité unconditionelle. Pour ceux-ci, on définit une notion de sécurité individuelle et on démontre que, s'il y a un numéro assez grand d'instances sécuritaires, alors l'avantage moyen d'accord avec la théorie de l'information est presque zéro. Cela suggère que la notion individuelle est plus raffinée que la version globale traditionnelle.

Dans la deuxième partie, on montre que les systèmes de compromis sécuritaires par rapport à la théorie de l'information peuvent être produits avec une composition d'un système chiffreur et d'un système d'authentification sécuritaires. On montre aussi que tout le système de compromis optimale peut être décomposé en un pair d'un système de chiffre et d'un système d'authentification sécuritaires, montrant ainsi que cette composition est la seule façon de faire des systèmes de compromis optimales.

Dans la troisième partie, on étudie la possibilité de généralizer l'analyse faite dans la première partie à des systèmes qui n'ont que des preuves de sécurité computationelle, par la comparaison de différentes notions d'entropie computationelle avec la valeur attendue de la complexité de Kolmogorov avec des limites de temps. L'objectif est de trouver un parallèle de la rélation éxistante entre l'entropie de Shannon et la valeur attendue de la complexité de Kolmogorov sans limites de temps. On montre qu'une moitié de cette relation se tient.

Contents

List of Tables

List of Figures

Index of Notation

$I(X;Y)$ the mutual information contained in random variable Y about X, 44

$I_K(x:y)$ the information contained in y about x, 67

$\tilde{I}(k:\mathbf{xy_w}|\mu)$ the average mutual information that a group of $q-1$ unqualified users can gain about the secret in a secret sharing scheme, 96

$K(x)$ prefix-free Kolmogorov complexity of x, 65

$K^t(x)$ the time-bounded Kolmogorov complexity of x, 70

$L_n(M;S)$ the expected length of a codeword for source S under the encoding scheme M, 137

$\mathbf{m}(\cdot)$ the universal distribution, 68

NP the class of decision problems solvable in nondeterministic polynomial time, 56

P the class of decision problems solvable in deterministic polynomial time, 56

$\Pr[A]$ the probability of event A occurring, 36

P_{d_i} deception probability of level i. For $i=0$, this is the probability of an impersonation attack, and for $i=1$ that of a substitution attack, 100

PPT the class of functions that can be computed in probabilistic polynomial time, 138

$\Pr[X=x, Y=y]$ the joint probability of random variable X taking value x and random variable Y taking value y, 39

$\Pr[X=x|Y=y]$ the conditional probability that random variable X takes value x given that random variable Y takes value y, 39

$SS(\mathcal{K}, \mathcal{S}, d, r)$ a secret sharing system, 75

$\mathbf{xy_{q-1}}$ the concatenation of some $q-1$ pairs of public and private share of a secret sharing scheme, 93

$\mathbf{xy_w}$ the concatenation of all public and private shares of a secret sharing scheme, 93

Chapter 1

Introduction

1.1 Cryptography through history

The need for secrecy has accompanied mankind from the beginning of civilization. As each society discovered writing, sooner or later it felt the need for secret writing. The techniques invented to reach this goal have developed through the ages into what we call today Cryptography. David Kahn, in 'The Codebreakers' ([Kah96]), gives a lengthy and exhaustive account of its development through history, and how it was mostly driven, until recent years, by the needs of state, military or political. He reports the emergence of steganography and cryptography in ancient times, in Greece, Mesopotamia, Egypt and India, and gives several accounts of the techniques used or simply theorized.

The first attempts at communicating secretly merely tried to conceal the existence of a message, which is the objective of steganography, but better than hiding a secret is to render it unintelligible, so that if the cover is somehow broken it will still remain inaccessible to the adversary. Such is the purpose of cryptography: to alter the text in a way that it becomes secret, or incomprehensible, except for a legitimate reader. Perhaps the most famous cryptographic user of Antiquity was Jules Caesar, who used to communicate in secret writing by replacing each letter by the one three places down in the alphabet. Even today, ciphers that consist of simply displacing an alphabet are called Caesar's ciphers. The counterpart to cryptography, the art of hiding messages, is cryptanalysis, that of retrieving the hidden meaning of a ciphered message. It was first systematically invented, and probably used, by the Arabs, and is documented in a book from the fifteenth century by Al-Qalqashandi which credited as its main source

a writer of the previous century, Ibn ad-Duraihim. Thus was born cryptology, the unification of cryptography and cryptanalysis in a single science.

Cryptology greatly developed in Europe from the Renaissance onwards. Different kinds of codes and ciphers were proposed and a steady progress in cryptanalysis maintained until Kerckhoffs, in the late nineteenth century, set down in writing the basic principles of military cryptology that are valid still today: the system must be unbreakable in practice, if not in theory; the security of the system should rest entirely on a secret key, and not on the details of the implementation: it is safe to assume the enemy knows all the details of its construction and operation, safe for the secret key. Furthermore, he reinforced that any given cryptographic system could be considered secure only after the excruciating examination of cryptanalysis.

It is this very concept of security, and especially that of theoretic security, that has eluded cryptographers through all the history of this discipline. No system was ever devised that could be called totally secure, or unbreakable; sooner or later, a cryptanalyst came and penetrated the layers of secrecy woven by each system. But there is one exception: Vernam's one-time pad, proposed during the first World War by Gilbert Vernam and eventually instituted in the German diplomatic service between 1921 and 1923 due to the work of Werner Kunze, Rudolf Schauffler and Erich Langlotz.

What makes the one-time pad secure is that it uses a totally random key that has the same length of the message that is going to be enciphered with the restriction that each key is used only once. Each character is enciphered by being added to the next character of the key. This system resists even a brute-force attack, for all the attacker learns from the ciphertext is the length of the message. But even though cryptographers had reached the conclusion that such a system was the only cryptographic system that could be theoretically secure, there was still no proof of it.

1.2 Perfect security

The proof came only in the middle of the twentieth century. In 1948, Claude Shannon inaugurated the discipline of information theory with his paper "A mathematical theory of communication" [Sha48], where the author defined the notion of entropy and showed that it corresponded to the amount of information associated with any given statistical event. One year later, in another paper entitled "Communication Theory of Secrecy Systems" ([Sha49]), he analysed secure communication and precisely defined what it means for a system to be secure, also showing that Vernam's scheme met this

definition. Not only that, any cipher system that is unconditionally secure must be isomorphic to the one-time pad.

From this moment on, cipher systems could be demonstrated secure or not, but there was one inconvenience: Shannon's proof showed that for a system to be secure, it had to use a key as long as the message, implying that to provide secure communication of a certain amount of text, at least as much text must be previously communicated in a completely secure fashion. This does not mean that these systems are useless, far from it. For top-level military or national secrets, none other are advised, but they entail a phase of pre-distribution of secrets that is sensitive and costly to perform, often requiring other security systems than the cipher system itself.

In simple terms, when one claims a certain cryptographic system is secure, this means that an attacker can not get more information from it than what is publicly available, according to the definitions of the system. This is a very intuitive view that conceals many parameters, for example, what the attacker can do to the system, how fast she can compute, what information she is allowed to have and what she should not be able to get. But one thing is fixed: the claim must hold when the attacker knows everything about the system's implementation, except the private information that is eventually held by the legitimate participants.

Meanwhile, cryptography began to be used in computer systems, which led to the emergence of other kinds of tasks that had to be securely carried on within untrusted or uncontrolled environments, or even in the presence of hostile and active users. Accordingly, cryptology swelled to include the study of many more different tasks than just secret communication, and today covers a whole array of different problems. Secrecy, however, is still at its heart. For some of these tasks, it has been possible to devise implementations with proofs of perfect security similar to that of the one-time pad. Information theory is the ubiquitous tool used in such proofs, and accordingly, perfect or unconditional security is also many times called information-theoretic security.

1.3 Computational Security

In the best of worlds, we would use only those cryptographic systems that we know are secure. However, as has been described above, the notion of security is not trivial and took the whole story of civilization to find out. Even now that we can prove some systems secure, we have found, to our dissatisfaction, that they require very long secret, perfectly random, keys to enjoy that security. Neither of these is

a practical requirement, meaning that they can not be carried out in situations of frequent and lengthy traffic, as for example, military communications on the field or the encipherment of all bank transactions occurring daily in the world. Unfortunately, ours probably is not the best world possible and we have to live with the restrictions imposed by reality. That includes accepting less than perfect cryptography.

Going back to Kerckhoffs's guidelines, we see that we can not use totally unbreakable systems in our daily life. So we turn to systems that can not be broken in practice. Unfortunately, we do not have any certainty in this field either. We have made fast progress since the 1980s, and we have defined what it means to be secure against practical adversaries. However, we still don't know any system that is secure according to such definition. There are several good candidates, for many different cryptographic problems and applications, but their security depends on the veracity of computational assumptions that we believe are indeed true, but we could not prove yet to be so.

These assumptions are statements about the computational power of a limited adversary, usually with restrictions in the maximum time available to perform its computations. Systems that are proven secure against computational adversaries, under these assumptions, are said to have provable security, or computational security.

This concept first appeared in a paper by Whitfield Diffie and Martin Hellman in 1976 [DH76], when they proposed to build cipher systems with a break-through idea: to use two keys instead of one. Each user of the system would have two keys, one of which she would publicize to every one who might send her a message. For this reason, this is called the public key. The other key must be kept secret. These two keys are such that it is very easy to compute a cipher text for each message with the public key. It is also easy to revert this encipherment if the private key is known, but unfeasible otherwise. But since there must be one specific private key for each public key, the system can not be unconditionally secure: given enough time, an adversary could test all possible private keys and find the only one that works. Although believed to be secure in practice, under Shannon's theory such systems are in fact extremely insecure.

The definitions of security vary according to the cryptographic task at hand, but for any given task, the definitions of information-theoretic and computational security are similar: they compare the probabilities an attacker has of succeeding at a given attack when she knows some extra piece of information, and when she simply guesses at random without knowing this extra information. The difference between these probabilities is called the advantage of the attacker. In information-theoretic systems, this must be 0 for all instances. In computationally secure systems, the probability

that this advantage is significant must be very low. It is in general not possible to outlaw the existence of bad instances, but we can design systems that, under the right assumptions, have only a negligible fraction of such bad cases.

1.4 Kolmogorov complexity

The cornerstone of information theory is the notion of entropy, defined by Shannon in his 1948 paper. It is a measure of the quantity of information present in a random source, or communicated by a random experiment. It is also a good measure of the total randomness present in a source. Since its inception, this quantity has been recognized as extremely useful and used in innumerable applications.

However, there are some inadequacies of entropy as a measure of randomness or information: first, it is defined for a given distribution, and therefore it gives a notion of the uncertainty over the result of that distribution, and in this sense of its randomness, but it can not say anything about the randomness of the individual objects in that distribution. This is reflected, for instance, in the following experiment: if we throw a coin in the air a hundred times and collect the results, we are as likely to get a string with a hundred zeroes as any other string, including one that looks random. Intuitively, though, we would be hard pressed to believe that a string of a hundred zeroes might be called random. It is, no doubt, a possible result of a random experiment, but intrinsically, the string contains no randomness at all. However, entropy can not capture this. Secondly, this same all-zeroes string does not contain much information: it is just the repetition, for a hundred times, of a single bit 0. If we compare it with a string of the same length full of 0s and 1s at irregular places, we must surely say that the latter must have more information in itself. But once again, this kind of difference between strings can not be determined by entropy alone.

These questions led to the development of an alternative notion to quantify information in the sixties, proposed independently by Solomonoff, Kolmogorov and Chaitin [Sol64, Kol65, Cha66]. This quantity is now known as Kolmogorov complexity and is defined as the length of the shortest program that can produce a given string. Unlike entropy, this quantity depends exclusively on the string, and not on the probability with which it is sampled from some given distribution. As such, Kolmogorov complexity measures the intrinsic information and randomness of a given string.

There is yet another important difference between entropy and Kolmogorov complexity: entropy is absolute, in that it considers the information contained in a

source irrespectively of the computational power available to the extractor of that information, and there is no obvious way to include these restrictions in its definition although, as will be seen later in this thesis, there have been proposals in that direction. Kolmogorov complexity is also absolute in the same sense, but it is much easier to consider the effect of time restrictions in this scenario. But there is an important drawback to Kolmogorov complexity: it is not computable, and even polynomial-time bounded Kolmogorov complexity takes an exponential time to compute.

So far, we have highlighted the differences between entropy and Kolmogorov complexity. Nonetheless, there are also important similarities between them, which will play a crucial role in this thesis: they both are measured in bits, their theories have similar properties, with formally equal inequalities, and it can even be shown that for a given distribution, the average of the Kolmogorov complexity of the elements in its support is asymptotically equal to the entropy of that distribution.

1.5 Individual security

The relation of entropy with Kolmogorov complexity and of the former with perfect security suggests the possibility that Kolmogorov complexity can be used in proofs of perfect security. Furthermore, since it can be easily adapted to include time-bound limits, it is at least a priori possible that it can be used in proofs of computational security, a field where entropy has little to say.

In this thesis, we explore the idea that Kolmogorov complexity can serve as a unifying notion within the field of cryptography, by working at the level of the instances instead of the distributions involved. The main objective is to provide proofs of security based on Kolmogorov complexity for public-key systems, while at the same time bringing the current theory of perfectly secure systems to work also under this quantity.

The approach we take is to consider and measure the security of individual instances. Instead of evaluating the advantage an adversary gains from the ciphertext in her probability at guessing a plaintext, we consider rather the amount of information about the plaintext that she gains from the ciphertext. This information is not dependent on the distribution used, but is intrinsic to the instance. Different distributions might give low or high probability to those instances in which the extra knowledge available to the attacker gives noticeable information about the secret. There is the potential, then, that a certain system is shown to be almost secure, or secure up to a certain parameter, for distributions that are not good enough according to entropy perfect

security proofs.

1.6 Organization of the Thesis

We begin in Chapter 2 by giving an introduction to several subjects that are needed to understand the thesis.

In Chapter 3, we define the notion of "individual security" using Kolmogorov complexity to quantize the information leaked in each instance. We believe this is a more realistic model of attack, since in practice, an attacker may not necessarily attempt to break all instances of a cryptosystem, but is likely to be willing to invest a lot of resources in breaking a single instance. We then analyse some cryptographic protocols, in order to determine which are the truly secure instances of the cryptosystem. We consider three basic types of information theoretically secure cryptographic systems: cipher systems, threshold secret sharing schemes, and authentication schemes. For each of these settings, we first give a Kolmogorov complexity based definition of security of an individual instance. Then we prove that security in the individual sense implies security in the traditional sense if the instances are still sampled according to the same distributions, by showing that if sufficiently many instances of a system are individually secure, then the system is also very nearly information theoretically secure. Finally, we identify the high-security instances of specific systems, using again properties derived from Kolmogorov complexity.

We note that under the notion of individual security, the perfectly secure systems that we analyse do have instances where the adversary gains a nonnegligible amount of information about the secret from the information she has access to and therefore the system does leak some few bits of information, on average, even despite the assurance that the attacker has 0 advantage according to the traditional definition. This is unavoidable unless we skew the distribution to give 0 probability to those instances. However, we have not analysed what would happen in this case to the information theoretic proof.

This distinction happens because the two concepts are fundamentally different, although related. The traditional point of view considers information as the inverse logarithm of the probability of an event, something that is not intrinsic to the event itself but to the distribution it is sampled from. The individual view, on the other hand, considers information as a property intrinsic to the string itself. It is a remarkable fact that these two definitions lead to similar results.

Since Kolmogorov complexity is not computable, we also consider the polynomial-time bounded version of Kolmogorov complexity and make an analysis based on it for the one-time pad. This measure naturally combines information and resource-bounded computation and can provide a natural way of bridging the gap between computational and unconditional security. In this work, it allows us to combine information theoretic security with a computable guarantee on the security of a particular instance of the system, conditional on an assumption on the resources available to the adversary.

Chapter 4 focuses on perfectly secure commitment systems and shows how these can be built from perfectly secure authentication and cipher systems. We also show that in the case of optimal commitment systems, this composition is the only way of building it, which means that optimal commitment systems are equivalent to a pair of a cipher system and an authentication system. Since both kinds are analysed in Chapter 3, this allows the individual analysis to extend over to commitment systems, although we have not made any specific work in that direction.

Chapter 5 provides a study of computational entropy in relation to polynomial time-bounded Kolmogorov complexity, with a view to extending the analysis of Chapter 3 to public-key systems in the future. We investigate a possible relation between time-bounded Kolmogorov complexity and some analogue of entropy for the computational setting. We achieve only one direction of this relation, namely upperbounding polynomial-time bounded Kolmogorov information by an affine function of Yao's effective entropy.

Finally, we conclude with some ending thoughts and open problems in Chapter 6.

Chapter 2

Preliminaries

This chapter gives an introduction to several subjects that are needed later in the thesis. We have tried to make the text self-contained so that the reader can find here all that is needed to understand later chapters. The following material was gathered from several sources, including [Sch96], [Kat02], [GB01], [Sha02], [AB07], [LV97], [Sha48], [Ant02], [Lap97], [AH97], [Cut80], [Gol01], [CT91] and several articles on Wikipedia.

2.1 Notation

This section covers some general guidelines with respect to the notation used throughout the thesis. More specific notations are introduced when they are first used, in particular in the remainder of this chapter.

The function log always denotes logarithm base 2, unless explicitly stated. For some finite set Σ, the notation Σ^* indicates the set of all string that can be formed by a finite concatenation of zero or more symbols from Σ. We call Σ an alphabet, and the strings formed from it are called words. The length of a word is the number of symbols it contains. The empty word has length 0 and is denoted by ϵ. A language L defined over some alphabet Σ is a subset of Σ^*.

Random variables are denoted by Roman capital letters, usually near the end of the alphabet, except when the letter used has some relation to the underlying concept (example, K for keys or C for ciphertext). We often represent sets by calligraphic type but also, on some occasions, by normal Roman capital letters. Definitions disambiguate if a letter represents a set or a variable. Lowercase Roman letters

usually represent binary strings from some set. If x represents a binary string, then x_i represents the i^{th} bit from the string and $x_{[a,\ldots,b]}$ represents the substring of x from bit a to b.

We use $|\cdot|$ to represent several things. When X is a set, $|X|$ represents the cardinality of X, while the notation $|x|$ represents the length of string x. If the argument of $|\cdot|$ is a number, then it represents its absolute value. The symbols $\lceil x \rceil$ represent the smallest integer that is not smaller than x, while $\lfloor x \rfloor$ represents the largest integer smaller than or equal to x.

2.2 Number theory

The holy grail of practical cryptography is what we call a perfect trapdoor one-way function. We give a formal definition in Section 2.8, but in simple terms, this is a function that is easy to compute and very hard to invert.

These two properties characterize one-way functions and any one of these would allow a sender to fabricate a secret that an attacker would be unable to find. But unfortunately, so would the legitimate receiver. The third necessary property for cryptography is the trapdoor aspect: the one-way function becomes easy to invert if some particular piece of information, the secret key, is known.

We don't yet know if one-way functions, let alone trapdoor one-way functions, exist, as their existence would imply $P \neq NP$. However, there are functions that are easy to compute but for which we have thus far been unable to find an efficient general inversion method. In practice, these are assumed to be trapdoor one-way functions. Most of the practical public-key cryptographic algorithms find their presumed one-way functions in number theory and so we present now a short introduction to this field.

When designing an algorithm for everyday use, it is not practical to consider that its inputs are of unlimited size. After all, any real computer has a limited memory and is unable to process instances that take more than a certain limit. For this reason, the algorithms we consider perform computations on numbers only up to a certain size. This is achieved by using modular arithmetic.

2.2.1 Modular Arithmetic

Modular arithmetic is done by reducing all the integers to equivalence classes by a congruence relation. For a fixed number $n > 0$, which is called modulus, a congruence class $[i]_n$ is defined to be the set of integers such that the remainder of their division by n is i. Given a positive number n, there is a unique way in which any number m can be written in the form $m = qn + r$ with $0 \leq r < n$, where all parameters are integers.

Thus, we define $[i]_n = \{m \in \mathbb{Z} : \exists j \in \mathbb{Z} \text{ s.t. } m = jn + i\}$ for i an integer such that $0 \leq i < n$, and the equivalence relation $a \equiv b$, called congruence modulo some number. We say a and b are congruent modulo n if they belong in the same congruence class, and write $a \equiv b \pmod{n}$. Equivalently, n divides $(a - b)$. We let $a \bmod n$ denote the actual operation of taking the remainder of a by n.

Clearly, for each $n > 0$ there are exactly n different congruence classes. Each congruence class has an infinite number of elements, so it is usual to represent it by its lowest non-negative member. That means that all congruence classes modulo n are represented by a single number ranging from 0 to $n-1$. These elements are also known as residues, and the set of all residues modulo n is denoted by $\mathbb{Z}_n$. Thus, for every integer a, its residue modulo n is $a \bmod n$. It is easy to check that $[a]_n + [b]_n = [a+b]_n$, $[a]_n - [b]_n = [a - b]_n$ and $[a]_n \cdot [b]_n = [a \cdot b]_n$. These make it possible to simplify computation, by representing all operands and intermediate results with only as many bits as needed for n. Modular arithmetic is associative, commutative and distributive, and therefore we can employ the usual rules of arithmetic. Division is another matter, though, as will be seen below.

2.2.2 Algebraic Structures

Definition 2.2.1 *A group G is a pair $G = (A, \cdot)$ where A is a set and $\cdot : A \times A \mapsto A$ is a total function that is associative, has a unique identity (also called neutral) element 1 and for every element $a \in A$ there is exactly one inverse $a^{-1} \in A$ such that $a \cdot a^{-1} = 1$. If $\cdot$ is commutative, the group is said to be abelian, or commutative.*

The above group is written in the so called multiplicative notation. A group can also be written in additive notation. In that case, the operation is represented by $+$, the identity element by 0 and the inverse by $-a$. The choice of notation is irrelevant, and is usually dependent on the context where the group will be used. The order of a

group is defined to be $|G|$.

A group is said to be cyclic if there is some element in G, say g, that generates all the other elements in the group by successive iterations of the operation of the group. The element giving origin to all others is called the generator of the group, or primitive element, and need not be unique.

An important example of a group is given by modular arithmetic: $(\mathbb{Z}_n, +)$ is a commutative group. Another important set is the associated $\mathbb{Z}_n^*$, for $n > 1$. This is the set of numbers in Z_n that have multiplicative inverse, or equivalently the set of residues modulo n that are coprime with n. The structure $(\mathbb{Z}_n^*, \cdot)$ is a finite multiplicative group. Its order is given by $\phi(n)$, where $\phi(n)$ is defined below.

Definition 2.2.2 (Euler's Totient Function) *The totient function of n, written $\phi(n)$, is the number of positive integers i with $i \leq n$ and $\gcd(i, n) = 1$.*

From the definition, $\phi(1) = 1$ and $\phi(p) = p - 1$ for any prime p. Furthermore, for prime p, $\phi(p^k) = (p-1)p^{k-1}$ and for m and n coprime, $\phi(mn) = \phi(m) \cdot \phi(n)$.

The group $\mathbb{Z}_n^*$ is cyclic if and only if $n = 2$, $n = 4$, $n = p^m$ or $n = 2p^m$, for some odd prime p and $m > 0$. For every element $a \in Z_n^*$, $a^{\phi(n)} \equiv 1 \pmod{n}$.

Theorem 2.2.3 (Fermat's Little Theorem) *For p prime and any $x \in \{1, \ldots, p-1\}$, $x^{p-1} \equiv 1 \pmod{p}$.*

An important consequence of this theorem is that for g and n such that g is a generator for $\mathbb{Z}_n^*$, $g^a \equiv g^{a \bmod \phi(n)} \pmod{n}$. In fact, writing $a = k\phi(n) + b$, we have $g^a = g^{k\phi(n)} \cdot g^b = (g^{\phi(n)})^k \cdot g^b = 1^k \cdot g^b = g^b$. This allows the possibility of reducing the exponent before operating, just like we are able to reduce summands and multiplicands in the other operations.

Definition 2.2.4 *A field F is a triple $F = (A, +, \cdot)$ such that $(A, +)$ and $(A \setminus \{0\}, \cdot)$ are abelian groups with identity elements 0 and 1 respectively and $\cdot$ is distributive over $+$.*

2.2.3 Galois Fields

A field that contains a finite number of elements is a finite field, also called Galois Field. Every finite field has order p^n, where p is some prime and n is a positive integer.

For every prime power p^n there is a finite field with that many elements and all finite fields of the same order are essentially the same field. This important fact allows us to identify finite fields by their size only, so it is customary to write $GF(p^n)$ to identify a particular Galois Field. The prime p is called the characteristic of the field. When the order of the field is a prime p, $GF(p)$ is isomorphic to $(\mathbb{Z}_p, +, \cdot)$, where $+$ and $\cdot$ are addition and multiplication modulo p with 0 and 1 as neutral elements.

However, when the order is a power of a prime, things are not so easy. The Galois Field $GF(p^n)$ is isomorphic to a field over polynomials on an abstract variable x of degree at most $n-1$ with coefficients in $\mathbb{Z}_p$. The equivalent notion, among polynomials, to prime numbers is that of irreducible polynomials. A polynomial is irreducible if it can not be written as the product of two polynomials of lesser degree and is primitive if it generates all the other polynomials in a field. A primitive polynomial is irreducible, and is guaranteed to exist for every $GF(p^n)$. We still have modular arithmetic in $GF(p^n)$, but the modulo now is a primitive polynomial of degree n for this field. We call this polynomial the field polynomial and denote it by $F(x)$. Then, any element $A(x)$ of this field is a residue modulo this polynomial, meaning that there must not be other polynomials $Q(x) \neq 0$ and $R(x)$ such that $A(x) = Q(x)F(x) + R(x)$ and $R(x)$ has degree less than that of $F(x)$.

Each polynomial can be represented as $A(x) = \sum_{i=0}^{n-1} a_i x^i$, where every $a_i \in \mathbb{Z}_p$, or more succinctly by $(a_{n-1}, \ldots, a_1, a_0)$. This can also be seen as a number in base p. Addition of polynomials is done one coefficient at a time, that is, $(a_{n-1}, \ldots, a_1, a_0) + (b_{n-1}, \ldots, b_1, b_0) = (a_{n-1} + b_{n-1}, \ldots, a_1 + b_1, a_0 + b_0) \pmod{p}$. To multiply two polynomials $A(x)$ and $B(x)$, we must first multiply the coefficients of $A(x)$ and $B(x)$ and then compute the remainder of the result by the field polynomial. This can be done using the algorithm of long division. These operations are very easy when $p = 2$, and an algorithm is given in [MV04].

2.3 Series

This section lists some results about series that are needed later.

Lemma 2.3.1 *For positive integer n, $\sum_{i=0}^{n} i2^i = 2^{n+1} \cdot (n-1) + 2$.*

Lemma 2.3.2 *For any integer a, $\sum_{i \geq a} \frac{i^2}{2^i} = \frac{(a+1)^2+2}{2^{a-1}}$.*

We end this section with an inequality that has found applicability in series, although in its most general sense it is a result about vectors in real or complex spaces.

Theorem 2.3.3 (Cauchy-Schwarz inequality) *If $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$ are two sequences of real numbers, then*

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

2.4 Probability Theory

Probability theory is concerned with the outcomes of random experiments.

Definition 2.4.1 *A finite probability space is a tuple (Ω, μ) where $\Omega = \{\omega_1, \ldots, \omega_k\}$ is a set of elementary events and μ is a probability function $\Omega \mapsto [0, 1]$ such that $\sum_{i=1}^{k} \mu(\omega_i) = 1$.*

A finite probability space describes the outcome of a random experiment by stating that an elementary event ω_i will occur with probability $\mu(\omega_i)$. For this reason, we often abbreviate $\mu(\omega_i)$ by p_i, the probability of the i^{th} elementary event.

In any experiment, we may consider events more complex than elementary events. For example, consider a deck of cards. We can consider the elementary event of drawing a specific card, or the more complex event of drawing a card from a given suit. We say that an event A over the space (Ω, μ) is any subset $A \subseteq \Omega$ and define its associated probability, denoted $\Pr[A]$, to be the sum of the probabilities of the elementary events belonging to A.

Definition 2.4.2 *Let $A \subseteq \Omega$ be an event over the probability space (Ω, μ). The probability that A occurs is*

$$\Pr[A] = \sum_{\omega \in A} \mu(\omega).$$

Clearly, for any event A, we have the basic properties:

1. $0 \leq \Pr[A] \leq 1$.

2. $\Pr[\Omega] = 1$.

3. $\Pr[\emptyset] = 0$.

4. $\Pr[\bar{A}] = 1 - \Pr[A]$, where $\bar{A}$ represents the event "not A".

5. $\Pr[A \cup B] = \Pr[A] + \Pr[B] - \Pr[A \cap B]$.

In fact, it is easy to prove the next lemma:

Lemma 2.4.3 (Union bound) *For every set of events $A_1, A_2, \ldots, A_n$,*

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] \leq \sum_{i=1}^{n} \Pr[A_i]$$

with equality if the sets $A_1, \ldots, A_n$ are mutually disjoint.

Definition 2.4.4 *The probability of events A and B occurring simultaneously, called the joint probability of A and B, is denoted by $\Pr[A \cap B]$. The probability of A occurring once B has occurred, called the conditional probability of A given B, is denoted by $\Pr[A|B]$ and defined to be*

$$\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]}.$$

Definition 2.4.5 *Two events A and B are said to be independent if $\Pr[A \cap B] = \Pr[A] \cdot \Pr[B]$. This is equivalent to $\Pr[A] = \Pr[A|B]$ or $\Pr[B|A] = \Pr[B]$.*

2.4.1 Random Variables

Usually, we are interested in something more than describing the outcome of an experiment: we may need to *quantify* it. For example, we might be interested in counting the number of cards drawn until a queen is drawn. To this purpose, we need random variables.

A random variable is a way to attribute a value to each possible event of the random experiment. This value can be taken from any conceivable set, and even be descriptive and not quantitative. However, we will only be interested in quantitative variables and so restrict this value to the set $\mathbb{R}$. We usually denote random variables by capital letters towards the end of the alphabet, like X, Y, S, T.

Definition 2.4.6 *A random variable X over probability space (Ω, μ) is a function $\Omega \mapsto \mathbb{R}$. We let $\Pr[X = x] = \sum_{i:X(\omega_i)=x} \mu(i)$ be the probability that X takes value x. For convenience, we often write $p_X(x)$ instead, and may even drop the X when the variable is understood. Usually we associate a probability function, for example ν, to a variable instead of a probability space and in this case we simply write $\nu(x)$.*

We usually call "probability distribution" to the function associated to probability $\Pr[X = x]$. In technical terms this is not very correct, as this function is properly called "probability function" which, in the case of discrete variables, is equal to the "probability density function". The "probability distribution function" is instead defined as $\mathcal{P}^*_X(x) = \sum_{y \leq x} \mathcal{P}_X(y)$. However, each function can be computed from the other, so we'll keep referring to p as the probability distribution. When needed to distinguish between the two, we'll do so carefully.

Definition 2.4.7 *The support set of a variable X is denoted by $[X]$ and is the set of the values the variable can take that have positive probability. Although X is a function that maps to set $\mathbb{R}$, for finite probability spaces the support set of X is finite.*

We use $x \sim X$ to denote the sampling of an element x according to distribution X, and $x \in_R S$ if we draw x randomly and uniformly from set S. Sometimes, for some random variable X, we use $x \in X$ as a shorthand to "x is in the support of X". We also use the notation $\Pr[X \in D]$ where X is a random variable and D is a set to signify the probability that a random value of X belongs to D, as a shorthand to $\sum_{x \in [X]} \Pr[x \in D]$.

Definition 2.4.8 *The expected value of a random variable X over (Ω, μ), denoted $E(X)$, is $E(X) = \sum_{i=1}^{k} \mu(\omega_i) X(\omega_i)$. When we need to highlight the distribution function μ, we write $E_{X \sim \mu}(X)$.*

An important theorem related to random variables is the following:

Theorem 2.4.9 (Linearity of Expectation) *For any two random variables X and Y, let $X + Y$ represent the variable induced by $(X + Y)(\omega) = X(\omega) + Y(\omega)$.*

For any random variable X and a real value α, let αX represent the variable induced by $\alpha X(\omega) = \alpha \cdot X(\omega)$.

Then, $E(X + Y) = E(X) + E(Y)$ and $E(\alpha X) = \alpha E(X)$.

Definition 2.4.10 *The variance of a random variable X, denoted by* $\mathrm{Var}(X)$ *or* σ_X^2, *is* $\mathrm{Var}(X) = E(X - E(X))^2 = \sum_{x \in [X]} p_X(x)(x - E(X))^2$.

The standard deviation is written σ_X *and defined* $\sigma_X = \sqrt{\mathrm{Var}(X)}$.

It is easy to check that the variance is also $\mathrm{Var}(X) = E(X^2) - (E(X))^2$.

On several occasions, we have to analyze different random variables defined over the same probability space.

Definition 2.4.11 *For two random variables X and Y defined over probability space (Ω, μ), the probability that the joint event (x, y) occurs is denoted* $\Pr[X = x, Y = y]$ *and is equal to* $\sum_{i:X(\omega_i)=x \text{ and } Y(\omega_i)=y} \mu(i)$.

We call the probability $\Pr[X = x, Y = y]$, *also denoted* $p_{XY}(x, y)$ *or even* $p(x, y)$, *the joint probability of X and Y.*

If we are given a joint probability distribution on X and Y we can easily recover the distributions on X and Y, with $p_X(x) = \sum_{y \in [Y]} p_{XY}(x, y)$ and respectively for Y. The distributions on X and Y are called the "marginal distributions". The reason is that the probability on the several pairs (x, y) can be written on a table with the x labelling the rows, for example, and the y labelling the columns. Then, if we compute the sum for each row and the sum for each column we get respectively $p_X(x)$ and $p_Y(y)$ for every x and y written on the margin of the table.

There are occasions when we need to analyse joint random variables that are independent and have the same distribution function. These variables are said to be independently identically distributed, which is often abbreviated to "i.i.d.". We also abbreviate the resulting joint random variable. For instance, the occurrence of k independent variables identically distributed to some variable X can be denoted as $X^k = \underbrace{X \times \cdots \times X}_{k}$. We can also define conditional distributions for random variables, as we did for events.

Definition 2.4.12 *The conditional probability that X takes value x when Y takes value y is defined* $\Pr[X = x | Y = y] = \frac{\Pr[X=x, Y=y]}{\Pr[Y=y]}$. *We also simplify the notation here to* $p_{X|Y}(x|y)$ *or just* $p(x|y)$.

Note that $\sum_{x \in [X]} p_{X|Y}(x|y) = 1$, so $p_X(\cdot|y)$ with y fixed is still a probability function. However, $\sum_{x \in [X], y \in [Y]} p_{X|Y}(x|y) = |[Y]|$, so $p_{X|Y}(\cdot|\cdot)$ is not a probability function.

Definition 2.4.13 *Two random variables X and Y are independent if for every pair (x, y), $p_{XY}(x, y) = p(x) \cdot p(y)$.*

2.4.2 Statistical Distance

In order to know if two distributions are close, we use the notion of statistical distance.

Definition 2.4.14 *Let P and Q be two distributions over the same probability space $(\Omega, \cdot)$. Let ε be a small real constant between 0 and 1. Then, P and Q are $\varepsilon-$close if*

$$\sum_{x \in \Omega} |P(x) - Q(x)| \leq 2\varepsilon.$$

Equivalently, we may say that for any event $S \subseteq \Omega$, $|P(S) - Q(S)| \leq \varepsilon$.

2.4.3 Useful inequalities

It is often necessary to estimate the probability with which a given random variable takes values above or below a certain point. There are three kinds of inequalities that are often used to this purpose.

Theorem 2.4.15 (Markov's inequality) *For any non-negative random variable X and $k > 0$, $\Pr[X \geq kE(X)] \leq \frac{1}{k}$.*

If the values that X can take are upper bounded, we can use Markov's inequality to give bounds in the opposite direction.

Theorem 2.4.16 *Let X be a random variable that takes values on $[0, 1]$. Then, for $k > 0$, $\Pr[X < kE(X)] \leq \frac{1-E(X)}{1-kE(X)}$.*

Markov's inequality can be applied for any non-negative random variable and requires only knowledge of its average. However, it does not usually give very good bounds. A stronger inequality that can be used when we also know the variance of the random variable is Chebyshev's inequality.

Theorem 2.4.17 (Chebyshev's inequality) *If X is a random variable with standard deviation σ, then for every $k > 0$, $\Pr[|X - E(X)| > k\sigma] \leq \frac{1}{k^2}$.*

The last inequality is very used in computer science, and in particular in computational complexity. It is stronger than the previous two, but it is also more stringent. It applies only to discrete random variables on $\{0, 1\}$.

Theorem 2.4.18 (Chernoff Bounds) *Let X be the sum of n independent discrete random variables X_i on $\{0, 1\}$, with $p_i = \Pr[X_i = 1]$. Let $0 < \delta < 1$. Then,*

$$\Pr[X > (1 + \delta)E(X)] \leq e^{-\delta^2 E(X)/3};$$
$$\Pr[X < (1 - \delta)E(X)] \leq e^{-\delta^2 E(X)/2}.$$

A special case is when the n variables X_i are all equally distributed. Then, $E(X) = np$, where $p = \Pr[X_i = 1]$ for all $1 \leq i \leq n$.

An alternative version (see [Gol01, Tre00]) is the following

Theorem 2.4.19 *Let X be the sum of n independent discrete random variables X_i on $\{0, 1\}$, with $p = \Pr[X_i = 1]$ for all i. Let $0 < \delta \leq p(1 - p)$. Then,*

$$\Pr\left[\frac{X}{n} - p > \delta\right] \leq e^{-\frac{\delta^2 n}{2p(1-p)}};$$
$$\Pr\left[\frac{X}{n} - p < -\delta\right] \leq e^{-\frac{\delta^2 n}{2p(1-p)}}.$$

2.5 Information Theory

Information theory is a mathematical model to deal with information. It quantifies information in function of its predictability or unlikeliness: the more surprising a given message is, the more information it contains. Intuitively, we can reason that if we receive a message that significantly alters the view we have of the world around us then that message must contain a lot of information. A message that, on the other hand, just confirms something that we are used to consider normal is pretty uninteresting and has little information. This is a generalization of the old principle of journalism that a news like 'Dog bit mailman' has no interest whatsoever, because it is more or less within dogs' usual role in the world to bite mail men's legs. The headline would be a lot more interesting, though, if it were reversed and read instead 'Mailman bit dog'. The underlying idea in this discussion is that the probability of the first event occurring is reasonably high, and so that event, when it happens, carries

with it little information, and of course the other way around for the second event. This is the implicit idea in Claude Shannon's work.

Shannon initiated the field of Information Theory with his paper of 1948 [Sha48], where he defined the core notion of entropy. Shannon describes a discrete information source as some device that can be, at any time, in one of a finite number of states. From each state, the device can make a transition to another state, possibly the same. The transition taken is chosen according to a predefined set of probabilities, and each transition produces a symbol of the output. In practice, this means that the source generates symbols from a finite set according to certain probabilities and the probability of each symbol depends on a finite number of previous symbols. Entropy is then a measure of the quantity of information the source produces, or better, the rate this source produces information at. The name entropy was chosen for similarity of the function introduced by Shannon to that of entropy in statistical mechanics.

A source as defined above can be modelled by a random variable that produces symbols from a finite set with a certain probability distribution. We can say the entropy of the source is the entropy of the associated random variable. As Shannon points out, what is important for information is not the actual content of the symbols produced but instead the probability with which they are produced. Events with less probability carry more information and events with high probability have less information. Consider an alphabet of symbols $\mathcal{X} = \{x_1, \ldots, x_n\}$ and a random variable X over this alphabet, with probabilities $p_i = \Pr[X = x_i]$. We can define the information produced by the source when it outputs each of these events as $I(x_i) = \log(1/p_i)$. This satisfies a number of desirable properties:

- The information associated to an event is non-negative.
- The more surprising events have more information.
- If two events x_1 and x_2 are independent, then their joint information is $I(x_1 \cup x_2) = I(x_1) + I(x_2)$.

The entropy of the random variable X is just the average $I(x_i)$ for all x_i.

Definition 2.5.1 *Let X be a random variable over support $\mathcal{X} = \{x_1, \ldots, x_n\}$ with probabilities $p_i = \Pr[X = x_i]$. The entropy of X is*

$$H(X) = -\sum_{x_i \in \mathcal{X}} p_i \log p_i$$

with the convention that $0 \cdot \log 0 = 0$, justified by $\lim_{x \to 0^+} x \log x = 0$.

As Shannon says in his paper, entropy is not only a measure of the information produced but also of the uncertainty associated with the outcome of a probabilistic source. The more symbols the source can produce, and the more evenly distributed they are, the more uncertain we are about the output of the source. This uncertainty is, after all, randomness, and therefore we can regard entropy as a measure of the randomness in a given source. If for some i we have $p_i = 1$, then there is no uncertainty in the experiment: the outcome is always the same. Therefore, this source is deterministic and the entropy must be 0, which indeed is verified by the definition. It can also be proved that, for each n, H is maximum for the distribution that assigns probability $1/n$ to all events, which corresponds to our intuitive notion of maximum uncertainty. Besides, the entropy in this case increases as the source alphabet increases. Notice that for a uniform distribution of events, $H(X) = \log |\mathcal{X}|$, where $\mathcal{X}$ is the set over which X is defined. It can also be seen that since H is the sum of non-negative quantities, then it must itself be non-negative. Therefore, $0 \leq H(X) \leq \log |\mathcal{X}|$.

Definition 2.5.2 (Joint Entropy) *Consider variables X and Y, with support sets $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, \ldots, y_m\}$ respectively. Let $p(i, j)$ be the occurrence of event $X = x_i, Y = y_j$. The joint entropy of these variables, written $H(X, Y)$ is defined as*

$$H(X, Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} p(i, j) \log p(i, j).$$

This implies that $H(X, Y) \leq H(X) + H(Y)$ with equality if and only if X and Y are independent. Similarly, we can define conditional entropy.

Definition 2.5.3 (Conditional Entropy) *Consider variables X and Y, with support sets $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, \ldots, y_m\}$ respectively. Let $p(i, j)$ be the occurrence of event $X = x_i, Y = y_j$. The conditional entropy of these variables, written $H(X|Y)$ is defined as*

$$H(X|Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} p(i, j) \log p(i|j).$$

Theorem 2.5.4 (Additivity of Entropy) $H(X, Y) = H(X) + H(Y|X)$.

From Definition 2.5.2, we can see that $H(X, Y)$ is symmetric. However, $H(X|Y)$ is not, since in general $H(X|Y) \neq H(Y|X)$. We have said above that $H(X, Y) = H(X) + H(Y)$ if and only if X and Y are independent. Coupled with the previous theorem, this gives another characterization of independent variables.

Corollary 2.5.5 *Consider two random variables X and Y. The variables are independent if and only if $H(X|Y) = H(X)$ or, equivalently, $H(Y|X) = H(Y)$.*

Since both $H(X)$ and $H(X|Y)$ are positive, it becomes also clear that $H(X|Y) \leq H(X)$, with equality only in the case of independence.

It is also possible to define the entropy of X when Y is known as $H(X|Y = y) = -\sum_{x\in\mathcal{X}} p(x|y) \log p(x|y)$ and to show that $H(X|Y) = \sum_{y\in\mathcal{Y}} p(y)H(X|Y = y)$.

The additivity of joint entropy can be generalized to conditional entropies:

Lemma 2.5.6 *For random variables X, Y and Z, $H(X, Y|Z) = H(X|Z)+H(Y|X, Z)$.*

This is used to derive a chain rule of operation for the joint entropy of several variables.

Theorem 2.5.7 *Let $X_1, \ldots, X_n$ be random variables, not necessarily independent. Then,*

$$H(X_1, X_2, \ldots, X_n) \leq \sum_{i=1}^{n} H(X_i).$$

Another important concept in information theory is that of the information between two variables. It can be defined as the distance between two distributions, but that amounts to the following definition.

Definition 2.5.8 *Let X and Y be two random variables. The amount of information contained in Y about X is*

$$I(X;Y) = \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x, y) \log \frac{p(x|y)}{p(x)}.$$

There is a simpler way to compute this quantity.

Theorem 2.5.9 *Let X and Y be random variables. Then,*

$$I(X;Y) = H(X) - H(X|Y).$$

Additivity of entropy easily shows that $I(X;Y) = I(Y;X)$, and for this reason, this quantity is also called *mutual information.*

2.5.1 Min-entropy

Min-entropy is a notion related to entropy that characterizes not the exact randomness, but instead a minimal amount of randomness that a source contains. It is a natural lower bound for entropy.

Definition 2.5.10 *The min-entropy of a random variable X, denoted $H_\infty(X)$, is defined as the minimum $-\log p(x)$ over all x.*

Theorem 2.5.11 *For any random variable X, $H_\infty(X) \leq H(X)$.*

2.5.2 Compressibility

Entropy and information theory enjoyed a great development in the study of codes and of the best way to encode messages to be transmitted over physical devices. One of the important results about entropy in communications theory is that the entropy of a source is a lower bound on the average codeword length of any code for that source. In this section, we briefly talk about codes and state this result in more formal terms.

A deterministic coding scheme for a source S over alphabet $\mathcal{X}$ is a pair of functions $e : \mathcal{X} \mapsto \mathcal{Y}$ and $d : \mathcal{Y} \mapsto \mathcal{X}$ such that for all $x \in \mathcal{X}$, $d(e(x)) = x$. We will restrict ourselves to binary codes here, that is, $\mathcal{Y} = \{0,1\}^*$.

A code is uniquely decodable if for any sequence of codewords $y_1 \ldots y_\ell$ there is only one sequence of source messages $x_1 \ldots x_\ell$ that could be encoded to it. For example, consider an alphabet of four source messages $\mathcal{X} = \{A, B, C, D\}$. If $e(A) = 1$, $e(B) = 10$, $e(C) = 11$ and $e(D) = 0$, then this code is not uniquely decodable. For message $ADBC$, we have codewords 101011, which is also the encoding of $BBAA$.

Being uniquely decodable is often not enough. A code might be uniquely decodable and yet be inefficient to decode. Consider the same source alphabet as before with the following encoding function: $e(A) = 0$, $e(B) = 01$, $e(C) = 011$ and $e(D) = 111$. This code is uniquely decodable, as 0 marks the beginning of a new codeword, except in the case of 111. If we see three 1s in a row we know for sure that is the codeword for D, for there is no other way we can have three 1s together. However, we might have more than three consecutive 1s and not immediately know where the codeword for D begins. For example, consider the text 0111110. We have to read the message until the last bit before we learn the first codeword. For illustration, consider the decodings for all the

possible truncated code messages: $d(0) = A$, $d(01) = B$, $d(011) = C$, $d(0111) = AD$, $d(01111) = BD$, $d(011111) = CD$ and $d(0111110) = CDA$.

In practice we want codes where each codeword can be uniquely decoded as soon as it is complete. Such codes are called instantaneous codes. They have the property that no codeword is the prefix of any other codeword and for that reason they are called prefix-free codes. There is an important inequality respecting prefix-free codes, known as the Kraft-McMillan inequality.

Theorem 2.5.12 (Kraft Theorem) *Let $l_1, \ldots, l_N$ be the codeword lengths for each of N codewords in a binary prefix-free code. Then,*

$$\sum_{i=1}^{N} 2^{-l_i} \leq 1.$$

Theorem 2.5.13 (McMillan Theorem) *Let $l_1, \ldots, l_N$ be the codeword lengths for each of N codewords in a uniquely decodable binary code. Then,*

$$\sum_{i=1}^{N} 2^{-l_i} \leq 1.$$

Since all uniquely decodable codes satisfy the Kraft-McMillan inequality, just like prefix free codes, it is always possible to turn a uniquely decodable code into an equivalent prefix-free code with equal codeword lengths. Thus, if there is an optimal uniquely decodable code, there is an optimal prefix-free code which achieves the same result. Therefore we can restrict the analysis to prefix-free codes. Now we give the result that establishes the importance of entropy in source-encoding.

Theorem 2.5.14 (Noiseless Coding Theorem) *Let L be the average codeword length of any binary prefix-free code for a source described by random variable X. Then, $H(X) \leq L$.*

Thus, the entropy of the source is the utmost limit to which we can compress, on average, the symbols of that source. On the other hand, although this will be of marginal interest for this thesis, it is also possible to prove the existence of almost efficient codes for any source.

Theorem 2.5.15 *For any random variable X, there exists a binary prefix code with average codeword length $L \leq H(X) + 1$. Such codes are called optimal codes.*

Examples of optimal codes are the Huffman encoding and the Shannon-Fano code. For the construction, see [CT91, GV04].

Theorem 2.5.16 ***[Shannon-Fano Code]*** *Let X be a random variable over set $\mathcal{X}$ with probability mass function f. Let $e(x)$ represent the encoding of source word x by the Shannon-Fano code. Then, $|e(x)| \leq \log 1/p(x) + 1$.*

2.6 Combinatorial Designs

The theory behind combinatorial designs has a long and interesting history. The properties of designs have been often used in the design of statistical experiments. Imagine a scenario where a number of subjects are drafted to test different varieties of a certain product. Since it is not practical to have all subjects test all the varieties, the researchers try to allocate a subset of them to each subject in a way that everyone tests the same number of cases and all varieties are equally tested. Designs have been used in cryptography to build unconditionally-secure authentication codes, but they have applications in many other fields. Colbourn, Dinitz and Stinson give an account of some of them in [CDS99].

In this section, we give an introduction to combinatorial designs, focused only on what we need later. This section is mostly based on [AH97].

Definition 2.6.1 ($t-$**designs**) *Let $\mathcal{S}$ be a set of size v whose elements are called points. Let $\mathcal{D}$ be a collection of subsets of $\mathcal{S}$ called blocks, all of them with exactly k elements. Let $b = |\mathcal{D}|$, $t \leq k < v$ and $\lambda > 0$.*

The pair $(\mathcal{D}, \mathcal{S})$ is a $t-(v, k, \lambda, b, r)$ design if every point occurs in exactly r blocks and every subset of $\mathcal{S}$ with exactly t points occurs in exactly λ blocks.

Most often, $t = 2$ and in this case, we call a design a Balanced Incomplete-Block Design, or BIBD.

An easier way to visualize designs is to cast them onto a table with the points labelling the columns and the blocks labelling the rows. Each cell in the table contains either 0 or 1, indicating whether a block contains a certain point. This matrix is called the incidence matrix of a design. This matrix is not unique, since it depends on the ordering of the points and the blocks, but that is not important for our purposes. From

the definitions above, each row contains exactly k 1s and each column has exactly r 1s. The parameters above are not totally independent. In fact,

Theorem 2.6.2 *If $(\mathcal{D}, \mathcal{S})$ is a $t-(v,k,\lambda,b,r)$ design, then $b = \frac{\lambda\binom{v}{t}}{\binom{k}{t}}$. Furthermore, each point occurs in exactly $r = \frac{\lambda\binom{v-1}{t-1}}{\binom{k-1}{t-1}}$.*

It is easy to see that in any such design $bk = vr$. This can be deduced from the above theorem, but an easier way to see it is to simply count the 1s in the incidence matrix first by columns, totalling vr, and then by rows, which gives bk.

Definition 2.6.3 *A design is said to be resolvable if its blocks can be partitioned into sets $\mathcal{P}_i$ called parallel classes, each with exactly v/k elements, such that the blocks in each parallel class form a partition of $\mathcal{S}$.*

A resolvable $1-(v,k,\lambda,b,r)$ design is called affine if for any two blocks B_1, B_2 belonging to different parallel classes, it happens that $|B_1 \cap B_2|$ is equal to k^2/v.

In such a design, we group together the blocks belonging in a certain parallel class, as in Table 2.1. We can see that in each class, no rows have 1s in the same positions and all points have 1 in exactly one row. Clearly, there are v/k blocks per parallel class.

1	2	3	4	5	6
1	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	0	0	0
0	0	0	1	0	1
1	0	0	0	1	0

Table 2.1: a resolvable 1-design

A design is called symmetric if the number of blocks is equal to the number of points, this is, $b = v$. The incidence matrix is therefore a square matrix, although it is not necessarily symmetric itself. The following lemma is taken from Theorem 2.14 in [AH97].

Lemma 2.6.4 *In a symmetric $t-(v,k,\lambda,b,r)$ design, any two blocks intersect in exactly λ columns.*

There are other kinds of combinatorial designs. The only other kind that we will use in this thesis is transversal designs.

Definition 2.6.5 *A transversal design $TD(k, n, \lambda)$ is a pair $(\mathcal{D}, \mathcal{S})$ such that $|\mathcal{S}| = k \cdot n$, the points in $\mathcal{S}$ can be divided into exactly k groups of n elements each, there are $\lambda \cdot n^2$ blocks, each of them containing at most one point from each group, and any pair of points from distinct groups occurs in exactly λ blocks.*

It is easy to see from the definitions that a transversal design is not a 2-design because two points from the same group are never contained in any block.

2.7 Computability Theory

The notion of computation is of fundamental importance in computer science, since it basically defines what computers are meant to do. The study of the nature of computability began in the 1930s, before the advent of computers, when many ideas were proposed to capture the right notion of what is computation and computability.

Intuitively, computation is the process that makes us produce some output from an input. An example is any set of successive rules to perform any given operation, like calculating the sum of two numbers or the detailed instructions to cook a cake from a set of ingredients. In the first case, we call these instructions an algorithm, but it is as much a recipe to obtain a desired result as our more culinary example.

Our intuition tells us that some function for which there is a well-defined set of rules to 'cook up' the result must be computable. These rules have to be finite and the result achievable in finite time. Computability theory, also known for historical reasons as recursive function theory, is concerned with characterizing functions and problems according to their being computable or not. And the first building stone of this theory is to define formally what a computable function is.

Computability is defined by some underlying computation model. The most used such model today is the Turing Machine, invented by Alan Turing in 1936 (see [Tur63]). Other models include the definition of functions built from composition, recursion and minimization of basic functions due to Gödel and Kleene ([Kle52]), Church's $\lambda-$calculus ([Kle52]), and URM-computable functions ([SS63]). All these computation models, and the associated notions of computability, have been proven to be equivalent in the sense that they all define the same class of computable functions. However, there

can be no mathematical proof that this class corresponds to all the functions that we can view as computable since that is, in essence, a subjective definition. In place of a theorem, we have what is known as Church-Turing thesis:

Definition 2.7.1 (Church-Turing thesis) *Any partial function that is computable by any algorithmic process is also computable by a Turing machine. Such functions are called 'Turing-computable'.*

In the above we could replace 'Turing machine' with any equivalent model of computation. This statement can not be proven, it simply is believed to be true based on the evidence that supports it: the equivalence of so many different computational models, the facts that we can write a computer program for any function in this class and that no one has ever presented an example of a function that would be accepted in the informal sense as computable while not being Turing-computable.

2.7.1 Turing machines

Definition 2.7.2 (Deterministic Turing machine)
A Turing machine $M(\Gamma, \Sigma, Q, \delta, q_0, q_{accept}, q_{reject})$ *is a device with a finite control, a reading/writing head and access to a work tape that is infinite in both directions. The tape is divided in cells and each cell can contain a single symbol from an alphabet of symbols* Γ*. One of these symbols indicates a blank space in the tape and does not belong to* Σ*. This is the alphabet of the inputs and is a subset of* Γ*. At the start of computation, the tape is filled with blank symbols, except for a finite number of contiguous cells at the starting position of the head, which is called the input.*

The finite control is defined by a finite set of states Q *and a transition function* $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$*. Time advances discretely and at any given point in time, the machine is in exactly one state of* Q *and the head is scanning exactly one cell of the tape. At the start of computation, the machine is in the initial state* q_0 *and the head is positioned over a designated cell of the tape. In a single step of computation, the machine looks up the symbol* a *under the head, matches it with the state* q *it is in and finds* $\delta(q, a) = (q', a', d)$*. Then, it writes symbol* a' *on the tape at the head's position and sets the internal state to be* q'*. Finally, it moves the head one cell to the left or right according to the direction* d*.*

There can be at most one entry in the transition function for each pair (q, a)*. This means the machine is deterministic. The computation stops, or halts, if the machine*

reaches a final state (an accepting or rejecting state), and is undefined if it never stops or there is no valid transition and the current state is not final. The output is whatever is left on the tape, starting from the initial position and until the first blank. Since, for $k \geq 1$, any tuple of $\mathbb{N}^k$ can be efficiently encoded to and decoded from $\{0,1\}^*$, a Turing machine defines a function $M : \mathbb{N}^k \mapsto \mathbb{N}$.

Alternatively, we may consider the machine does not have final states, and instead it stops if it reaches a state without valid transitions, accepting the input if the output is 1, *and rejecting it if the output is* 0.

This is a basic definition of Turing machine. It can be shown that an equivalent definition is obtained if more tapes are added, discriminating between a read-only input tape, a write-only output tape and read-write work tapes, if the tapes are infinite only in one direction or using a different finite alphabet. For each of these characteristics, for every Turing machine with the characteristic, there is another machine without it that simulates the first, and vice-versa, and these simulations can be done in time polynomial on that taken by the first machine. This makes this model quite robust and allows us to restrict ourselves to binary alphabets and adopt the tapes configuration more suitable to each problem.

Definition 2.7.3 *A function $f : \mathbb{N}^k \mapsto \mathbb{N}$ is said to be Turing-computable if there is a Turing machine M such that for every x, $M(x)$ halts and outputs b if and only if x is in the domain of f and $f(x) = b$.*

If a Turing machine M halts for all inputs, we say the function computed by M is total. Functions computed by Turing machines are said to be recursive. We are often concerned with the computation of predicates, this is, functions where the result is either 0 or 1. These can be interpreted as boolean functions, and we say a predicate P holds for x if $P(x) = 1$ and it does not hold if $P(x) = 0$.

Definition 2.7.4 *A predicate P is decidable, or recursive, if there is a Turing machine M such that:*

- *$M(x) = 1$ if $P(x)$ holds.*
- *$M(x) = 0$ if $P(x)$ does not hold.*

A predicate P is undecidable if it is not decidable.

Definition 2.7.5 *Consider a recursive predicate P and a Turing machine M for P. For any x, if $M(x) = 1$, we say M accepts x; if $M(x) = 0$, we say M rejects x. The set of x which M accepts is denoted by L_M and is said to be the language that M recognizes.*

Equivalently, a language L can be associated to a Turing machine M that accepts exactly those elements in the language.

Definition 2.7.6 *A predicate P is semi-decidable, partially decidable or recursively enumerable (often abbreviated to r.e.), if there is a Turing machine M such that:*

- *$M(x) = 1$ if $P(x)$ holds.*
- *$M(x)$ does not halt if $P(x)$ does not hold.*

In what follows, we will often use the word "problem" instead of "predicate".

There are several variants of Turing machines that are important in computational complexity and used in this thesis. These are listed below.

Definition 2.7.7 (Non-Deterministic Turing machine) *A non-deterministic Turing machine (NDTM) is a Turing machine with the following difference: the transition function is replaced by a mapping that may associate more than one transition to a given pair of source state and tape symbol. In this case, the machine chooses the next state arbitrarily from the possible transitions.*

Definition 2.7.8 *A NDTM is said to accept a language L if for every $x \in L$ there is a computation path that accepts x. Conversely, for every $x \notin L$, all computation paths for x must reach the rejecting state. The machine must be defined for all inputs.*

Definition 2.7.9 (Probabilistic Turing machine) *Formally, a probabilistic Turing machine (PTM) is a nondeterministic Turing machine with two transition functions. At each step, the machine tosses a fair coin and chooses one or the other.*

A probabilistic Turing machine can be seen as having an extra read-only tape with a succession of truly random bits which it uses at each step to decide its next configuration. We usually refer to these bits as the internal coins of this machine.

A way to think about nondeterministic machines is that if the machine accepts an input then it is able to guess the correct path that accepts that input. In other words, the machine accepts an input x if there is at least one computation path for x that ends in an accepting state. On the contrary, with probabilistic machines we are specifically concerned with the probability that an accepting path is chosen for each x.

Definition 2.7.10 (Relativized Turing machines) *A relativized Turing machine is a Turing machine which has access to free knowledge, by being allowed to ask questions to an oracle A. Basically, this oracle is able to determine in a single step if a certain instance x belongs to language A or not.*

The machine has an extra oracle tape on which it can write a string x. Then, it enters a query state and in the next step jumps to a YES state if $x \in A$ or a NO state otherwise. Relativized Turing machines can be deterministic or nondeterministic, and are written M^A, where M is the basic machine and A is the oracle it has access to.

2.7.2 Universal machines

A remarkable and important fact of the theory of computability is that the set of computable functions is enumerable, i.e., each computable function can be associated to several natural numbers and each natural number uniquely identifies a computable function. This follows from the existence of an effective enumeration for Turing machines and the fact that a given Turing machine computes one given function.

We can enumerate the set of Turing machines by defining an encoding scheme that attributes to each machine a single unique number. A machine is completely defined by its set of states and transition function, and since these are finite, there is a way to effectively code and decode these onto the natural numbers. Such encodings are given in [LV97] and [AB07]. Therefore, each machine has a tag m and these tags can be ordered lexicographically. We attribute to each m an index i indicating its position in the ordering and accordingly number the machines as $T_1, T_2, \ldots$ where T_i is the machine corresponding to the i^{th} number in the list. This process a bijection from the set of Turing machines onto the naturals.

The effective reversal of such encodings gives us the notion of Universal Turing machine. This is a machine U that receives as input a pair (i, x), builds the machine T_i from i and then simulates it on input x. Therefore, $U(i, x) = T_i(x)$. Based on this, we can view a universal machine U as a general-purpose computing device that

receives two inputs, p and x, where we view the first as a program for U and x as the data for that program, just like a general computer. The existence of a universal Turing machine has fundamental implications in computability, namely, the existence of incomputable problems. Consider the following:

Definition 2.7.11 (Halting problem) *Given a description i of a Turing machine and an input x, is there an algorithmic process to determine if $U(i, x)$ stops?*

It can be proven by diagonalization that such a problem is undecidable. To see this, consider an enumeration of all Turing machines $T_1, T_2, \ldots$ and that we have a Turing machine M such that $M(x) = 1$ if and only if $U(x, x)$ stops and $M(x) = 0$ if and only if $U(x, x)$ does not stop. Now, consider a second Turing machine N that is defined like this: $N(x) = 0$ if $M(x) = 0$ and $N(x)$ does not stop if $M(x) = 1$. Clearly, if M exists so does N. However, if N exists, then for any x, if $N(x)$ is defined then $U(x, x)$ does not stop and so $T_x(x)$ is not defined; if $N(x)$ is undefined, then $U(x, x)$ stops and so $T_x(x)$ is defined. This clearly makes N different from every Turing machine T_x on at least one input, so N can not exist and therefore so can't M.

But determining whether $U(x, x)$ stops is a particular case of the halting problem, and if this case can not be decided, neither can the halting problem be. Therefore, the halting problem is undecidable.

2.8 Computational Complexity Theory

The main concern of complexity theory is to study problems according to the resources necessary to solve them and how these increase with the length of the input. The principal objective is to identify those problems that are intractable, this is, that can not be solved in practice even with much more powerful computers than what we currently have. However, it must be noted that all the problems analyzed in complexity theory are still theoretically solvable: all of them are computable, although they may require too many resources for their computation to be feasible.

The resources typically considered are time, space, randomness and communication, but in this thesis, and in this introduction, we will focus only on time. Complexity theory tries to categorize problems in a hierarchy of classes defined by the amount and type of resources necessary to solve their problems. In order to make this separation, there must be a standard model of computation, and the most basic model is the

Turing machine. Other models have been used, like circuits, but we do not use them here. For the Turing machine, the time a computation takes is the number of steps executed by the machine, and the space taken by the computation is the number of tape cells scanned. Instead of measuring the exact time or space needed, we ignore additive and multiplicative constants and focus on the asymptotic behaviour of the resource bounds, viewed as functions of the input size. To properly analyze these, we use a set of specialized notations.

2.8.1 Asymptotic notation

Let f and g be functions from $\mathbb{N}$ to $\mathbb{R}$. We say that

- $f \in O(g)$ if there exists a constant $c > 0$ such that $f(n) \leq c \cdot g(n)$ for every sufficiently large n (f does not grow faster than g).
- $f \in \Omega(g)$ if there exists a constant $c > 0$ such that $f(n) \geq c \cdot g(n)$ for every sufficiently large n (f grows at least as fast as g).
- $f \in \Theta(g)$ if both $f \in O(g)$ and $f \in \Omega(g)$ (f and g grow at the same rate).
- $f \in o(g)$ if for every constant $c > 0$ and for every sufficiently large n, $f(n) \leq c \cdot g(n)$ (f grows slower than g).
- $f \in \omega(g)$ if for every constant $c > 0$ and for every sufficiently large n, $f(n) \geq c \cdot g(n)$ (f grows faster than g).

Usually, these functions are defined such that the constant c can be either negative or positive, but we mostly use them to replace positive quantities, and so we have specified c to be always positive in order that the resulting expressions conform more easily to intuition. Consequently, asymptotic terms sometimes have negative signs. We frequently abuse notation a bit and write, for example, $f(n) = O(n)$ instead of $f(n) \in O(n)$.

2.8.2 Basic Complexity Classes

A complexity class is a set of recursive functions which can be computed within a specified bound of a given resource in a given model of computation. In this thesis, we will consider decision problems only. These can be conveniently represented by the

set of elements x for which the boolean function describing the problem outputs 1. Consequently, we will view complexity classes as sets of languages instead of sets of functions.

Definition 2.8.1 (DTIME) *Let $t : \mathbb{N} \mapsto \mathbb{N}$ be some function.* $\mathbf{DTIME}[t(n)]$ *is the class of all languages decidable in time $O(t(n))$ by some deterministic Turing machine.*

Definition 2.8.2 (NTIME) *Let $t : \mathbb{N} \mapsto \mathbb{N}$ be some function.* $\mathbf{NTIME}[t(n)]$ *is the class of all languages decidable in time $O(t(n))$ by a nondeterministic Turing machine.*

From these definitions, we define the following important classes, **P**, **NP** and **EXP**.

Definition 2.8.3 (P) $\mathbf{P} = \bigcup_{c \geq 1} \mathbf{DTIME}[n^c]$.

Definition 2.8.4 (NP) $\mathbf{NP} = \bigcup_{c \geq 1} \mathbf{NTIME}[n^c]$.

Definition 2.8.5 (EXP) $\mathbf{EXP} = \bigcup_{c \geq 1} \mathbf{DTIME}[2^{n^c}]$.

The class **P** is the set of those problems that can be solved within time polynomial on the size of its input. Traditionally, it has been viewed as the definition of tractable or feasible computation, even though it may allow exponents so high that in practice the problem would still be unfeasible. Still, this criterium affords a clear and robust separation between tractable and intractable problems. The class **NP** has another equivalent definition given by the following theorem:

Theorem 2.8.6 *For every $L \subseteq \{0,1\}^*$, $L \in \mathbf{NP}$ if there exists a polynomial $p : \mathbb{N} \mapsto \mathbb{N}$ and a polynomial-time deterministic Turing machine M such that for every $x \in \{0,1\}^*$,*

$$x \in L \Leftrightarrow \exists y \in \{0,1\}^{p(|x|)} s.t. M(x,y) = 1.$$

This definition highlights an important characteristic of **NP** problems: if an element is in the language, there is a fast proof for that. However, we may not be able to quickly determine if a given element belongs in the language or not. This is the difference between verifying a solution for a problem and computing that solution. It is clear that $\mathbf{P} \subseteq \mathbf{NP}$, but we still don't know if $\mathbf{NP} \subseteq \mathbf{P}$. This is the most famous problem of complexity theory. **NP** covers the class of problems that can be solved by an exhaustive

search and test of a solution space where the test can be applied in polynomial time, which intuitively seems much more difficult than deterministically computing the right answer without having to look over all possible solutions. However, in three decades of research, we have not yet been able to prove or disprove this statement.

Sometimes, having access to true randomness can make a problem easier. Many problems for which we do not know deterministic feasible solutions are solved in practice with access to randomness and probabilistic arguments that show the solution occurs with high probability. This kind of computation is modeled by probabilistic complexity classes. We only need one of them for our purposes, **BPP**.

Definition 2.8.7 (BPTIME) *Let L be some language over $\{0,1\}^n$. Let $\chi_L(x) = 1$ if $x \in L$ and $\chi_L(x) = 0$ otherwise. We say $L \in$ **BPTIME**$[t(n)]$ if there is a probabilistic Turing machine M (equivalently, a probabilistic algorithm A) running in time $O(t(n))$ regardless of its random choices such that for all $x \in \{0,1\}^n$,*

$$\Pr[M(x) = \chi_L(x)] \geq \frac{2}{3}$$

where the probability is over the random coins used by M.

Definition 2.8.8 (BPP) $\mathbf{BPP} = \bigcup_{c \geq 1} \mathbf{BPTIME}[n^c]$.

The constant $\frac{2}{3}$ in the above definition can be replaced by any other constant larger than $1/2$ and at most 1, since for every PPT-algorithm A, it is possible to create another PPT algorithm A' that on input x executes $A(x)$ for a polynomial number of times with fresh randomness in each execution and decides by majority.

Theorem 2.8.9 (Error Reduction) *Suppose there is a PPT algorithm A and a polynomial $p(\cdot)$ such that, for some constant $c > 0$, A solves language $L \subseteq \{0,1\}^n$ with probability at least $\frac{1}{2} + \frac{1}{n^c}$ for every instance. Then, for every constant $d > 0$, there is a PPT algorithm A' that solves every instance of the same language with probability at least $1 - 2^{-n^d}$.*

Proof: Let A' be an algorithm that on input x, runs $A(x)$ for $k(n) = \frac{n^{2c+d}}{2}$ times, each with independent randomness. Then, A' will output the bit returned by most executions of A. Let $A^j(x)$ mean the j^{th} execution of $A(x)$. Since each execution uses fresh randomness, the several $A^j(y)$ are independently identically distributed (i.i.d.). Define the random variable X_j to be 1 iff $A^j(x) = \chi_L(x)$ and 0 otherwise. Then, the

several X_j are discrete binary variables with average $\frac{1}{2}+\frac{1}{n^c}$. We apply Chernoff bounds to estimate the probability of failure of $A'(x)$. Let $X = \sum_{j=1}^{k(n)} X_j$ be the number of successes among all the executions of $A(x)$. Then, the answer of A' will be correct if $X \geq \frac{1}{2}k(n)$. Let $p = \frac{1}{2} + \frac{1}{n^c}$ and note that $E(X) = \frac{1}{2}k(n) + k(n)\frac{1}{n^c}$. By the Chernoff bounds (Theorem 2.4.19),

$$\begin{aligned}
\Pr\left[X < \frac{1}{2}k(n)\right] &= \Pr\left[\frac{X}{k(n)} - p < -\frac{1}{n^c}\right] \\
&\leq \exp\left(-\frac{\frac{k(n)}{n^{2c}}}{2\left(\frac{1}{2}+\frac{1}{n^c}\right)\left(\frac{1}{2}-\frac{1}{n^c}\right)}\right) \\
&= \exp\left(-\frac{\frac{n^{2c+d}}{2n^{2c}}}{\frac{1}{2}-\frac{2}{n^{2c}}}\right) \leq e^{-n^d} \leq 2^{-n^d}. \qquad (2.1)
\end{aligned}$$

□

All these classes are said to be uniform, because we are considering a single machine that solves all instances of a problem. We can instead consider a family of different machines for each size of the input instance, and we call these classes non-uniform. We will only refer one non-uniform class, **P/Poly**.

Definition 2.8.10 (P/Poly) *Consider a polynomial $a : \mathbb{N} \mapsto \mathbb{N}$. Then,* **P/Poly** *is the class of all languages L such that there is a deterministic Turing machine M and a sequence of advice strings $\{\alpha_n\}_{n\in\mathbb{N}}$ with $\alpha_n \in \{0,1\}^{a(n)}$ such that $x \in L \Leftrightarrow M(x, \alpha_{|x|}) = 1$, $a(n)$ is polynomial on n and M runs in polynomial time on the size of x.*

2.8.3 Pseudorandomness

The complexity classes defined above are characterized by the amount of resources needed to solve the worst-case instances. This is not appropriate for cryptography, where we require that a cryptographic system be secure for most of its instances. Breaking a cryptographic algorithm is akin to solving a difficult problem. It is clearly not enough to assume that the worst-case instance is difficult, for it could happen that the majority of all other instances were easily breakable. This reasoning led to a necessary complexity-theoretic notion for cryptography, the one-way function, suggested for the first time in [DH76].

It is convenient to define negligible functions before continuing.

Definition 2.8.11 *A function $\varepsilon(n)$ is said to be negligible if for all polynomial p and large enough n, it happens that $\varepsilon(n) < \frac{1}{p(n)}$.*

Definition 2.8.12 (Strong one-way function) *A function $f : \{0,1\}^* \mapsto \{0,1\}^*$ is called strongly one-way if it easy to compute and the probability that any polynomial-time bound algorithm A inverts it is negligible. Formally,*

1. *There exists a deterministic polynomial-time Turing machine M such that on input x, $M(x) = f(x)$.*
2. *For every probabilistic polynomial-time Turing machine M and every positive polynomial $p(n)$, for sufficiently large n,*

$$\Pr[M(f(x), 1^n) \in f^{-1}(f(x))] < \frac{1}{p(n)},$$

where the probability is taken over all the instances x chosen uniformly over $\{0,1\}^n$ and the internal random choices of M.

The value n is the length of the intended output of M. The reason the argument 1^n is included and written in unary is to allow the machine to run in time polynomial on the combined lengths of its input and output. This definition avoids considering as one-way a function f that simply shrank its input so much that time polynomial on $|f(x)|$ would not be enough to print x. When $|f(x)| = |x|$, this argument is redundant.

In this definition, f is not assumed to be injective, but in many cases, particularly in number theoretic functions, this is the case and each $f(x)$ has only one inverse.

Definition 2.8.13 (Weak one-way function) *A function $f : \{0,1\}^* \mapsto \{0,1\}^*$ is called weakly one-way if it easy to compute and the probability that any polynomial-time bound algorithm A errs in computing an inverse is not negligible. Formally,*

1. *There exists a deterministic polynomial-time Turing machine M such that on input x, $M(x) = f(x)$.*
2. *There exists a polynomial $p(n)$ such that for every probabilistic polynomial-time Turing machine M, for sufficiently large n,*

$$\Pr[M(f(x), 1^n) \notin f^{-1}(f(x))] > \frac{1}{p(n)},$$

where the probability is taken over all the instances x chosen uniformly over $\{0,1\}^n$ and the internal random choices of M.

The difference between these two definitions is only on the second condition, and it seems at first glance that they characterize two very different groups of functions. Strong one-way functions are weak by definition, but Yao ([Yao82]) showed that the existence of weak one-way functions implies the existence of strong one-way functions. More, it is possible to create a strong one-way function from any weak one-way function. We still don't know if one-way functions exist at all, as their existence implies $\mathbf{P} \neq \mathbf{NP}$.

A notion that has turned out to be intimately related to one-way functions is that of pseudorandom generators. This notion was introduced in two papers, [Yao82, BM84], that actually gave different definitions of pseudorandomness. They both define pseudorandomness for an infinite sequence of distributions of strings. For Blum and Micali, the defining characteristic of a random distribution is that its output in successive iterations must be unpredictable from the output of all previous iterations.

Definition 2.8.14 (Blum-Micali) *Let $\{g_n\}$ be a polynomial-time computable family of functions with $g_n : \{0,1\}^n \mapsto \{0,1\}^m$ and m is a function of n with $m > n$. Then, this family is $(\varepsilon(n), t(n))-$unpredictable if for every probabilistic Turing machine A running in time $t(n)$ and every large enough n,*

$$\Pr[A(g(x)_{[1,\ldots,i]}) = g(x)_{i+1}] \leq \frac{1}{2} + \varepsilon(n),$$

where the probability is over the instances x uniformly sampled from $\{0,1\}^n$, $i \in \{1,\ldots,n\}$ and the internal coins of A. If for every fixed k, $\{g_n\}$ is $(1/n^c, n^k)-$unpredictable for every $c > 1$, then we say it is unpredictable by polynomial-time algorithms.

Yao's definition considers instead that a pseudo-random distribution must be indistinguishable from a true random distribution by computationally bounded Turing machines. He formalizes this by stating that g is a pseudorandom generator if any polynomial-time Turing machine can not properly distinguish whether its input comes from a true uniformly random distribution or instead is the result of the application of g over strings coming from a true, but shorter, random distribution.

Definition 2.8.15 (Yao) *Let $\{g_n\}$ be a polynomial-time computable family of functions with $g_n : \{0,1\}^n \mapsto \{0,1\}^m$ where m is a function of n with $m > n$. Then, this family is a $(\varepsilon(n), t(n))-$pseudorandom generator if for every probabilistic Turing machine M running in time $t(n)$ and every large enough n:*

$$\left| \Pr_{y \in_R \{0,1\}^m}[M(y) = 1] - \Pr_{x \in_R \{0,1\}^n}[M(g(x)) = 1] \right| \leq \varepsilon(n).$$

We say g is a pseudorandom generator, in short, if $\varepsilon(n) \leq \frac{1}{p(n)}$ for all polynomial $p(n)$ and $t(n)$ is any polynomial. We say the output of g is pseudorandom.

Yao also proved the equivalence of both definitions. The technique he used became known as the hybrid argument and is a centrepiece of cryptography today.

Definition 2.8.16 (Computational indistinguishability) *Two distributions X and Y over $\{0,1\}^n$ are $(\varepsilon(n), t(n))$- indistinguishable if for any deterministic Turing machine M running in time $t(n)$ with access to an auxiliary truly random input string r of size up to $t(n)$,*

$$\left| \Pr_{y \sim Y, r \sim U_{t(n)}} [M(y,r) = 1] - \Pr_{x \sim X, r \sim U_{t(n)}} [M(x,r) = 1] \right| \leq \varepsilon(n).$$

We say X and Y are computationally indistinguishable if $\varepsilon(n) \leq \frac{1}{p(n)}$ for all polynomial $p(n)$ and $t(n)$ is any polynomial.

This definition turns out to be an analog of statistical distance. Consider the notion of computational distance from [BSW03].

Definition 2.8.17 (Computational distance) *Let C be a class of functions $f : S \mapsto \{0,1\}$. The computational distance of distributions X and Y over S, with relation to C, is*

$$\mathrm{cdist}_C(X,Y) = \max_{f \in C} |E[f(X)] - E[f(Y)]|.$$

This can be written more explicitly, when f is deterministic, as $\mathrm{cdist}_C(X,Y) = \max_{f \in C} |\sum_{x \in S} f(x) \cdot p_X(x) - \sum_{x \in S} f(x) \cdot p_Y(x)|$. To better see the relation between this notion and statistical distance, note that f actually selects a subset A from S. Besides, this subset is a language in C, since its characteristic function is in C. So,

$$\begin{aligned}
&\max_{f \in C} \left| \sum_{x \in S} f(x) \cdot p_X(x) - \sum_{x \in S} f(x) \cdot p_Y(x) \right| = \\
&\max_{A \subset S \cap C} \left| \sum_{x \in A} p_X(x) - \sum_{x \in A} p_Y(x) \right| = \\
&\max_{A \subset S \cap C} |X(A) - Y(A)|,
\end{aligned}$$

which is a generalization of statistical distance restricting the sets we can investigate. Now, we show how this is related to Definition 2.8.16. Yao's definition can be written as

$$\max_{M \in \mathbf{PTM}_p} \left| \Pr_{y \sim Y, r \sim U^{t(n)}}[M(y,r) = 1] - \Pr_{x \sim X, r \sim U^{t(n)}}[M(x,r) = 1] \right| \leq \varepsilon(n),$$

where $\mathbf{PTM}_p$ is the set of all probabilistic Turing machines running in polynomial time. Each Turing machine actually realizes a probabilistic function $f(x)$, that returns 1 for each x with a certain probability depending on x. We can replace the class of probabilistic Turing machines by some class C for probabilistic boolean functions, so we can rewrite the above as $\max_{f \in C} \left| \sum_{s \in S} \Pr[Y = s] \cdot f(s) - \sum_{s \in S} \Pr[X = s] \cdot f(s) \right| \leq \varepsilon(n)$ or, since $f(s)$ is probabilistic, as $\max_{f \in C} |E[f(Y)] - E[f(X)]| \leq \varepsilon(n)$, this is, a statement on the computational distance.

It is easy to show that the existence of pseudorandom generators implies the existence of one-way functions. In essence, if $g(x)$ is a pseudorandom generator, then it must also be a one-way function. If not, then inverting it and finding the seed would distinguish $g(x)$ from a truly random string of the same size. The opposite direction was established in [HILL99]. In the same paper, the authors introduced a notion of computational entropy that basically generalizes Yao's definition of pseudorandomness, by saying that a distribution has computational entropy at least b if it can not be computationally distinguished from a distribution with Shannon entropy b. Then, a distribution over size n strings is pseudorandom if it has computational entropy at least n, i.e. it is computationally indistinguishable from the uniform distribution.

2.9 Kolmogorov complexity

Shannon's theory gives an adequate measure of the randomness present in a stochastic source. This randomness is the consequence of the outcome of the source being, a priori, unpredictable. Consider now the simplest of random sources, a fair coin tossed over and over, and a transcript of its results over a finite number of iterations: a binary string. Can we say that such string is random, since it is the output of a random source?

Perhaps the first natural answer is "yes", because we are anticipating the source to behave in a certain way. We hardly expect to see in the result string sequences like 00000000000000000000 or 01010101010101010101. We simply don't think these strings could be the output of a random process. The idea of randomness suggests total unpredictability and independence of each bit relative to the previous ones, and these

strings certainly don't show such traits. Compare with a similar size string obtained by tossing a coin: 01010110111011100111. However, all of these strings are quite possible outcomes of the experiment "toss a fair coin 20 times" and moreover, they are all equally likely.

What this suggests is that we have an intuitive idea of the properties of random experiments and the frequency these occur with. We do not expect to see an all-0 string because there is only one such possible outcome. On the other hand, the last string seems acceptable because there is no clear pattern in it and so we are unable to distinguish it from many other similar strings. Kolmogorov complexity seeks to quantify the randomness in a finite string according to the existence or not of regularities that can be identified by algorithmic processes: a string must be random if it can not be identified by some significantly shorter description. To avoid possible paradoxes, we have to be careful in the descriptions we allow, and we restrict these to computable functions.

Definition 2.9.1 (Kolmogorov complexity) *Let M be some fixed Turing machine. The Kolmogorov complexity of a binary string x relative to M is the length of the shortest input that instructs M to produce x. If no such string exists, it is defined to be ∞.*

$$C_M(x) = \begin{cases} \min_p\{|p| : M(p) = x\}, & \text{if there is such a } p \\ \infty, & \text{otherwise.} \end{cases}$$

This definition is dependent on the choice of M and therefore it is not robust. However, the existence of universal Turing machines solves this problem. To distinguish from another variant of Kolmogorov complexity that we will be using, $C(x)$ is also called *plain* Kolmogorov complexity.

Theorem 2.9.2 (Invariance Theorem) *Let U be a fixed universal Turing machine. Then, for any other Turing machine M, $C_U(x) \leq C_M(x) + c_M$, where c_M is a constant depending on M but not on x.*

This theorem allows us to fix some universal Turing machine for once and for all. We thus omit the subscript from $C_U(x)$ and write $C(x)$ instead. This definition can be extended to the case where we give some additional input to the program computing x.

Definition 2.9.3 (Conditional Kolmogorov complexity) *Let U be a fixed universal Turing machine. Then for any binary strings x, y, the Kolmogorov complexity*

of x given y, or x conditioned on y, is $C(x|y) = \min_p\{|p| : U(p, y) = x\}$. For the empty string ϵ, $C(x|\epsilon)$ is defined to be equal to $C(x)$.

We can also define the complexity of two concatenated strings. Usually, we are more interested not in the complexity of xy but in that of a reversible coding of x and y, that is, some representation of both strings from which we can derive each of them. This coding is represented by $\langle x, y\rangle$ and we often omit the angle brackets when they are understood from context.

Kolmogorov complexity is uncomputable, and can not be lower bounded by recursive, unbounded non-decreasing functions. However, we can give an upper bound to it.

Theorem 2.9.4 (Basic properties) *For some constant c that does not depend on x,*

- $C(x) \leq |x| + c$.
- *For any partial computable function ϕ, $C(\phi(x)) \leq C(x) + c$.*
- $C(x|y) \leq C(x) + c$.

Definition 2.9.5 *A string x is said to be $c-$incompressible if $C(x) \geq n - c$, and it is Kolmogorov-random if it is $0-$incompressible.*

Such strings can be seen to exist by a counting argument, as will be shown later.

2.9.1 Prefix Kolmogorov Complexity

Kolmogorov complexity as defined above does not satisfy a desirable property, namely that for every pair of strings x, y, $C(\langle x, y\rangle) \leq C(x) + C(y) + c$. It seems natural that given descriptions for x and y we have a description for $\langle x, y\rangle$. However, this description would have to be input to a Turing machine U as a unique string, and U would have to split it into a program for x and another for y. The problem is we do not know where one ends and the other begins. Prefix-free Kolmogorov complexity does not have this problem. It is defined as plain Kolmogorov complexity, with the exception that the input to the Turing machine must be a codeword from a prefix-free domain. In other words, for any two programs p and q for which this machine halts, neither is a prefix of the other.

In the example above, to separate x from y we have to add some extra information. Suppose p is a description for x and q is a description for y. We may precede pq by the length of p. But this new string has to be itself delimited. One way to do this is to write each bit of $|p|$ repeated, with the exception that the last bit is coded as 01 if it is a 1 or 10 if it is a 0. Let $n = |p|$. This way, $|p|$ is coded in $2n \leq 2\log p + 2$ bits and the whole description for $\langle x, y \rangle$ takes at most $C(x) + C(y) + 2\log C(x)$ bits.

Definition 2.9.6 (Prefix-free Kolmogorov complexity) *Let U be a universal prefix-free Turing machine. The prefix Kolmogorov complexity of a binary string x is the length of the shortest prefix-free program that makes U produce x.*

$$K(x) = \min_p \{|p| : U(p) = x\}.$$

The above definition works because, as it happens for regular Turing machines, there is also an effective enumeration of prefix-free Turing machines and a universal prefix Turing machine. Consequently, the invariance theorem holds for $K(x)$ as well.

We can define the conditional prefix Kolmogorov complexity as before, merely replacing the universal machine by a prefix one. From now on, we will use exclusively $K(x)$ and refer to it simply as Kolmogorov complexity. We state the basic properties for $K(x)$.

Theorem 2.9.7 (Basic properties) *For any binary strings x and y and some constant c that does not depend on x,*

- $K(x) \leq |x| + 2\log|x| + c$.
- *For $n = |x|$, $K(x|n) \leq |x| + c$.*
- *For any partial computable function ϕ, $K(\phi(x)) \leq K(x) + c$.*
- $K(x|y) \leq K(x) + c$.

Prefix-freeness ensures sub-additivity of Kolmogorov complexity.

Theorem 2.9.8 *For any binary strings x and y, $K(x, y) \leq K(x) + K(y) + c$.*

As before, we can define random strings based on $K(x)$.

Definition 2.9.9 *A string x is said to be $c-$incompressible if $K(x) \geq n - c$.*

Kolmogorov complexity is an accurate measure of the inherent randomness in a given string, independently of its being or not the outcome of any random experiment. It also quantifies the exact amount of information necessary to produce a certain string, and in this sense it is a quantification of the information contained therein. An important result in the theory of Kolmogorov complexity is the existence of random strings.

Theorem 2.9.10 (Incompressibility Theorem) *For each constant c, the number of binary strings x of size n with complexity $K(x) \leq n - c$ is at most 2^{n-c}.*

Proof: By a counting argument, there are at most 2^{n-c} prefix-free programs of size up to $n - c$, so at most 2^{n-c} strings can have complexity up to $n - c$. □

This result holds also for the conditional case and can be strengthened to the following result, which we will not proved here (see [LV97] for a proof).

Theorem 2.9.11 *For each constant c, the number of binary strings x of size n with complexity $K(x) \leq n - (c - K(n))$ is at most 2^{n-c}.*

2.9.2 Two part description

If we know that x belongs to some set A, then we may find that x is easier to describe. All we have to do is to give a description of A and then an index to the position x occupies in some ordering of A. A itself may be very easy to describe, for instance, the set of all $n-$bit strings, which essentially has complexity $O(\log n)$, or very complex, for example if $A = \{x\}$, which has complexity $K(x)$ that may be very large. The description of x's position within A is an index i such that $1 \leq i \leq |A|$ and can be described in at most $\log |A|$ bits. Therefore,

Theorem 2.9.12 *Let A be a partial recursive set such that $x \in A$. Then, $K(x) \leq K(A) + \log |A| + O(1)$, where $K(A)$ represents a minimal description of the characteristic function of A.*

If A is a partial recursive set, then there is a partial recursive function ϕ that enumerates A, and this can be described by a program with length equal to a constant c_ϕ independent of x. Therefore, in this case, we can write $K(x) \leq \log |A| + O(1)$.

2.9.3 Mutual Information

We can define the information in a string y about x in a way analog to that of information theory:

Definition 2.9.13 *For any two binary strings x and y, the information y contains about x is defined as $I(x : y) = K(x) - K(x|y)$.*

Contrary to information theory, this quantity is not even asymptotically symmetric. To make it so, we need another definition, which we denote by $K_c(x)$, due to Chaitin. Let x^* denote the first shortest program, in a lexicographic ordering, that produces x. Then, let $K_c(x) = K(x)$ and $K_c(x|y) = K(x|y^*)$. This implies $K_c(x, y) = K(x, y)$ and $K(x|y^*) = K(x|y, K(y))$, since $K(x^*) = K(x, K(x))$.

Theorem 2.9.14 (Addition Theorem) *For any two binary strings x and y,*

$$K_c(x, y) = K_c(x) + K_c(y|x) + O(1) \Leftrightarrow$$
$$K(x, y) = K(x) + K(y|x^*) + O(1).$$

This result is credited to Gács (1994) in [GV04]. Now, we can redefine the information in y about x:

Definition 2.9.15 (Mutual information) *For any two binary strings x and y, the information y contains about x is defined as*

$$I_K(x : y) = K(x) - K(x|y^*).$$

This quantity is called "mutual information" because it is symmetric. Due to the addition theorem, $I_K(x : y) = K(x) - K(x|y^*) = K(x) - K(x, y) + K(y) = I_K(y : x)$. We use this definition in the rest of the thesis. It is worth to note the following result:

Theorem 2.9.16 *Let x, y be binary strings. Then, $I(x : y) \leq K(x)$ and $I(x : y) \leq K(y)$.*

Proof: Consider $I(x : y) = K(x) - K(x|y)$. Since $K(x|y) \geq 0$, the first inequality follows easily.

To prove $I(x : y) \leq K(y)$, note that $K(x) \leq K(y, x) \leq K(y) + K(x|y)$. This implies $K(x) - K(x|y) \leq K(y) \Leftrightarrow I(x : y) \leq K(y)$. □

This is also true for $I_K(x : y)$ with practically the same proof. The important point is that this does not depend on symmetry of information and means the information x has about y can not be more that the total information x has within, and of course can not exceed the quantity of information there is to know about y.

There are generalizations for the conditional case. The following theorem and definition are taken from [Gác88].

Theorem 2.9.17 (Conditional addition theorem) *For any three binary strings x, y and z, $K(x, y|z) = K(x|z) + K(y|x, K(x|z), z) + O(1)$.*

Definition 2.9.18 (Conditional mutual information) *For any three binary strings x, y and z, $I_K(x : y|z) = K(x|z) + K(y|z) - K(x, y|z)$.*

We sometimes use an extended notation of x^*, namely $\langle x, y\rangle^*$, justified by this lemma:

Lemma 2.9.19 *For arbitrary strings x, y, z, the following is true up to an additive constant: $K(x, y|z^*) - K(y|z^*) = K(x|\langle y, z\rangle^*)$.*

Proof: $K(x|\langle y, z\rangle^*) = K(x, y, z) - K(y, z) = K(z) + K(x, y|z^*) - K(z) - K(y|z^*) = K(x, y|z^*) - K(y|z^*)$. □

2.9.4 Universal measure

Definition 2.9.20 *A function $f : \mathcal{A} \mapsto [0, 1]$ is called a probability measure over a space $\mathcal{A}$ closed for countably many unions and intersections if $f(\emptyset) = 0$, $f(\mathcal{A}) = 1$ and for any countable (possibly infinite) sequence of mutually disjoint sets $E_n \subseteq \mathcal{A}$ we have $f(\bigcup_{n=1}^{\infty} E_n) = \sum_{n=1}^{\infty} f(E_n)$. It is called a semi-measure if for a sequence $\{E_n\}$ of disjoint sets $\sum_n f(E_n) < 1$. We call the measure discrete if the set S is discrete.*

It is possible to define a universal recursively enumerable discrete semi-measure over the set of all recursively enumerable discrete probability semi-measures.

Definition 2.9.21 (Universal semi-measure) *Consider an enumeration of all r.e. discrete semi-measures $P_1, P_2, \ldots$. We define the universal r.e. discrete semi-measure* **m** *by*

$$\mathbf{m}(x) = \sum_{n \geq 1} 2^{-K(n)} P_n(x).$$

This yields $2^{K(n)}\mathbf{m}(x) \geq P_n(x)$ for all x and every enumerable discrete semi-measure P_n. To simplify, we let $P = P_n$, $K(P) = K(n)$ and write $2^{K(P)}\mathbf{m}(x) \geq P(x)$.

If we toss n fair coins in the air, the probability that this produces any $n-$bit string is $1/2^n$. But if we consider this string as the input to a Universal Turing machine, then the probability that a given x is generated is instead the sum over the probabilities of a program for x being generated. This quantity is known as a priori probability of x.

Definition 2.9.22 (A priori probability) *The universal a priori probability of a string x is defined as*

$$Q_U(x) = \sum_{p,U(p)=x} 2^{-|p|},$$

where U is a fixed prefix-free universal Turing machine and the sum runs over all programs of any size that halt producing x in this machine.

It can be seen that $Q_U(x)$ is a probability measure because $\sum_x Q_U(x)$ runs a set of prefix-free programs and by Kraft's inequality we get that $\sum_x Q_U(x) \leq 1$. To make it sum to 1, we simply attribute the remaining probability to some extra word that denotes an infinite computation, for example, some conventional way to represent ∞.

It is a common principle in science to favour the simplest explanation that justifies a set of data. This is known as Occam's razor. In algorithmic terms, this is the explanation with lowest Kolmogorov complexity, so we can define a probability measure.

Definition 2.9.23 (Algorithmic probability) *The algorithmic probability of x is defined as $R(x) = 2^{-K(x)}$.*

There is an important relation between these three notions, known as the Coding Theorem.

Theorem 2.9.24 (Coding Theorem) *There is a constant c such that for every x,*

$$-\log \mathbf{m}(x) = -\log Q_U(x) = K(x)$$

with equality up to c.

In the rest of this thesis, we let $\mathbf{m}(x) = 2^{-K(x)}$ and call it universal distribution.

2.9.5 Time-bounded Kolmogorov Complexity

A way to make Kolmogorov complexity computable is the introduction of resource bounds. By giving the reference universal Turing machine a limit on the time or the space it can use, we can search for all programs within those bounds that produce x and thus compute the resource-bounded Kolmogorov complexity of x. In this thesis, we will give our attention only to time-bounded Kolmogorov complexity.

Definition 2.9.25 (Time-bounded Kolmogorov complexity) *Fix a universal prefix-free Turing machine U and a time-constructible*[1] *function $t(n)$. We define the time bounded Kolmogorov complexity of a binary string x as*

$$K^t(x) = \min_p\{|p| : U^t(p) = x\},$$

where by $U^t(p)$ we mean that U receives p in its input tape and computes it until $t(|x|)$ steps have passed. If the computation has not ended by then, U stops outputting whatever is on the tape at that moment.

Most of the simplest properties of $K(x)$ also hold for $K^t(x)$. Namely, $K(x) \leq K^t(x) \leq |x| + 2\log|x| + O(1)$. In fact, $K^t(x)$ approaches $K(x)$ as $t(|x|)$ grows. We can similarly define a conditional version $K^t(x|y)$ and the incompressibility theorem also holds. It is not known if time-bounded symmetry of information holds for $K^t(x)$ when t is a polynomial: if it holds then one-way functions do not exist. The paper [LR05] studies this question and proposes a new version of Kolmogorov complexity for which polynomial-time symmetry of information holds. This is discussed in Section 2.9.6. It is also possible to define, by analogy, a quantity $\mathbf{m}^t(x) = 2^{-K^t(x)}$. This is a probability function, and we can compute the corresponding distribution function, or cumulative probability, $\mathbf{m}^{*t}(x) = \sum_{y \leq x} \mathbf{m}^t(y)$. The function $\mathbf{m}^t(x)$ is computable in time $t(n)2^{n+1}$ by simulating all programs of size up to n in time $t(n)$. Using this, we can compute $\mathbf{m}^{*t}(x)$ in time $t(n)2^{2n+2}$ by computing all intermediate sums. Better methods of computing these functions are still not known, thus we can't say if $\mathbf{m}^t(x)$ or $\mathbf{m}^{*t}(x)$ are $t(n)-$time computable. However, we can state a dominance theorem relating these functions to $t(n)-$time computable functions.

Definition 2.9.26 *Denote by $\mathcal{P}^{*t}$ the class of all probability distributions P^* that are computable in time $t(n)$.*

[1] A function f is time-constructible if some Turing machine computes $f(n)$ within $f(n)$ steps.

Theorem 2.9.27 *Let $t(n)$ be a polynomial and $t'(n) = nt(n)$. For all P with corresponding $P^* \in \mathcal{P}^{*t}$, there is a constant c_P independent of x such that for all x $2^{c_P}\mathbf{m}^{t'}(x) \geq P(x)$, for $c_P = K^{t'}(P) + O(1)$.*

When we introduce time bounds, the length of the shortest program that produces a string x may not be the equal to that of shortest program that accepts only x. Without time-bounds, this difference does not exist, but in the time-bound realm it has led to two different definitions: the one we gave above and that of Distinguishing Complexity. We do not need the latter in this thesis and so do not mention it further.

2.9.6 CAM Complexity

In [LR05], the authors defined another notion of time-bounded Kolmogorov complexity for which polynomial-time symmetry of information does hold. This is a variant where the program has the same computing ability as the class[2] AM, where the computation is done in probabilistic polynomial time with bounded error (Arthur), with the help of nondeterminism (Merlin).

Definition 2.9.28 (CAM complexity) *Let U be a fixed universal nondeterministic Turing machine that takes as input a program p, a data string y and a random auxiliary string r. Then for any string $x, y \in \Sigma^*$, for a polynomial t, the t-time-bounded AM decision complexity $CAM^t(x|y)$ is the length of the smallest program p such that*

1. *for all $r, U(p, y, r)$ runs in at most $t(|x|+|y|)$ steps,*
2. *with probability at least 2/3 over the choice of r, $U(p, y, r)$ has at least one accepting path, and $U(p, y, r) = x$ on all of the accepting paths.*

Theorem 2.9.29 (Polynomial-time bounded symmetry of information, [LR05]) *For any polynomial time bound t, there exists a polynomial time bound t' such that*

$$K^t(x, y) \geq CAM^{t'}(x) + CAM^{t'}(y|x) - O(\log^3(|x| + |y|)).$$

[2] AM is the class of decision problems for which the answer can be verified by an Arthur-Merlin protocol: Arthur is a probabilistic polynomial time verifier who, based on the input, sends a challenge and a series of random coins to an unbounded prover Merlin and then outputs an answer "yes" or "no" based on Merlin's response, being that Arthur has at least 2/3 probability of answering correctly even if Merlin cheats.

It can also be proved that $CAM^{2t'}(x,y) \leq CAM^{t'}(x)+CAM^{t'}(y|x)+c$. We can use the programs for x and y given x to compute the pair (x,y) as we do for normal complexity. The tricky part is to ensure that this computation has at least one accepting path with probability at least $2/3$. If we simply run the program to obtain y first and at the end of each accepting path run the program to obtain x from y, then the probability of having no accepting paths in the first phase is at most $1/3$, and in the second is at most $1/3$ times whatever the probability is of having an accepting path on the second phase. These bounds are not good enough, so we need a trick.

To compute (x,y), U receives a program q for y and a program p for x given y plus some instructions to execute p and q like this: it parses q from its input string and launches two copies of $U(q,r)$. Each of the copies has an accepting path with at least $2/3$ probability, so the pair of machines has no accepting path with probability at most $1/9$. At the end of each accepting path, the master machine launches two copies of $U(p,y,r)$. Once again, the probability that there's no accepting paths between each pair of machines is at most $1/9$. Thus, the probability that there are no accepting paths outputting x for this master machine is at most $1/9+1/9 = 2/9 \leq 1/3$. The execution time on a complete accepting path is at most $t'(|x|+|y|)+t'(|y|) \leq 2t'(|x|+|y|)$. So, we have a CAM description for (x,y).

2.9.7 Relation with Shannon Entropy

There is a very close relation between Shannon's entropy and Kolmogorov complexity: entropy is the expected value of Kolmogorov complexity for computable distributions. In this sense, Kolmogorov complexity is a sharper notion than entropy. The following theorem follows from Corollary 4.3.2 and Theorem 8.1.1 in [LV97]. It is given here since we could not find it elsewhere in the literature.

Theorem 2.9.30 *Let X,Y be random variables over $\mathcal{X},\mathcal{Y}$. For any computable probability distribution $\mu(x,y)$ over $\mathcal{X}\times\mathcal{Y}$, $0 \leq \left(\sum_{x,y}\mu(x,y)K(x|y) - H(X|Y)\right) \leq K(\mu)+O(1) = O(1)$, since $K(\mu)$ does not depend on x.*

Proof: Let μ_Y and μ_X be the marginal probability distributions over $\mathcal{Y}$ and $\mathcal{X}$ respectively. Similarly, denote by $\mu_{X|Y}$ and $\mu_{Y|X}$ the conditional probability distributions. For the first inequality, $H(X|Y) = \sum_y \mu_Y(y)H(X|Y=y)$ which is at most $\sum_y \mu_Y(y)\sum_x \mu_{X|Y}(x)K(x|y) = \sum_{x,y}\mu(x,y)K(x|y)$ where the inequality follows from the Noiseless Coding Theorem (Theorem 2.5.14), since y is a fixed string.

For the second direction, Corollary 4.3.2 in [LV97] states that $K(x|y) \leq \log 1/\mu_{X|Y}(x|y) + K(\mu) + O(1)$. Therefore,

$$\begin{aligned}\sum_{x,y} \mu(x,y)K(x|y) &= \sum_y \mu_Y(y) \sum_x \mu_{X|Y}(x|y)K(x|y) \\ &\leq \sum_y \mu_Y(y) \sum_x \mu_{X|Y}(x|y) \log 1/\mu_{X|Y}(x|y) + K(\mu) + O(1) \\ &= \sum_{x,y} \mu(x,y) \log 1/\mu_{X|Y}(x|y) + K(\mu) + O(1) \\ &= H(X|Y) + K(\mu) + O(1).\end{aligned}$$

□

Note that $K(\mu)$ is a constant c_μ depending only on the (computable) conditional probability of X and Y, but not on the particular value of x [Cha75].

An analogous result for mutual information holds.

Theorem 2.9.31 ([Gác88, GV04]) *Let X, Y be random variables over $\mathcal{X}, \mathcal{Y}$. For any computable probability distribution $\mu(x,y)$ over $\mathcal{X} \times \mathcal{Y}$, $I(X;Y) - K(\mu) \leq \sum_{x,y} \mu(x,y) I_K(x : y) \leq I(X;Y) + 2K(\mu)$. When μ is given, then $I(X;Y) = \sum_{x,y} \mu(x,y) I_K(x : y|\mu) + O(1)$.*

2.10 Cryptography

2.10.1 Information-Theoretic Security

The base of information-theoretic proofs of security is the concept of entropy and the related notion of mutual information of random variables. A cryptographic system often requires that the knowledge of some public information available to an attacker does not help her in finding anything about some secret information. This is usually formalized by considering a random variable over the sets of public and private information, for example, X and Y respectively, and requiring that their mutual information be 0, or $I(X : Y) = 0 \Leftrightarrow H(X|Y) = H(X)$. This means the attacker gains no advantage by knowing the ciphertext, no matter which one this is.

We give an example with a cipher system (although there are many more kinds of systems with information-theoretic security). The advantage of the attacker is a function of the difference in the likelihood of each plaintext occurring for a given

ciphertext and its a priori likelihood of occurring: considering (x, y) is a pair of plaintext x and corresponding ciphertext y, the advantage is $\log 1/p(x) - \log 1/p(x|y)$.

This technique, unfortunately, is not applicable to public-key systems. For example, in the widely used RSA, each public-key uniquely determines the corresponding private key, which means that the entropy of the plaintext when one knows the ciphertext and the public key is exactly zero. But this is because entropy does not take in consideration the difficulty of making that computation.

2.10.2 Cryptographic Algorithms

This section gives an introduction to the cryptographic algorithms studied in Chapters 3 and 4.

2.10.2.1 Symmetric Cipher Systems

The purpose of a cipher system is to conceal from an eavesdropper the messages that one person sends to another.

Definition 2.10.1 *A cipher system is a tuple denoted* $CP(\mathcal{P}, \mathcal{C}, \mathcal{K}, f(k,p))$ *where* $\mathcal{P}$ *is the alphabet of plaintext messages,* $\mathcal{C}$ *is the alphabet of ciphertext messages and* $\mathcal{K}$ *is the alphabet of secret keys. For each* $k \in \mathcal{K}$*, there is an encrypting function* $f_k : \mathcal{P} \mapsto \mathcal{C}$ *with* $f_k(p) = f(k,p)$ *that is injective and defined for all* $p \in \mathcal{P}$*. For each* k*, the decrypting function is* $g_k : \mathcal{C} \mapsto \mathcal{P}$ *defined as* $g_k = f_k^{-1}$.

A cipher system is unconditionally secure if the random variables $P, K, C = f(K, P)$ satisfy $H(P) = H(P|C)$.

2.10.2.2 Secret Sharing Systems

Secret sharing systems are useful when a secret needs to be split among several parties such that a minimum number of them is necessary to reconstruct the secret. The first implementations for this kind of system were given in 1979 independently by Shamir ([Sha79]) and Blakley ([Bla79]). Both their systems are threshold secret sharing schemes.

Definition 2.10.2 (Threshold Secret Sharing) *Let q, w be positive integers, $q \leq w$. $SS(\mathcal{K}, \mathcal{S}, d, r)$ is a (q, w)* threshold scheme *if $d : \mathcal{K} \rightarrow \mathcal{S}^w$ produces w shares of a key $k \in \mathcal{K}$ in such a way that any q participants can compute the value of k by using the reconstruction function r, but no group of $q - 1$ participants can do so.*

The attribute "threshold" is used above to denote that any group of users that contains more than a certain number of different shares, the threshold, is able to recover the secret. This is in opposition to more complex secret sharing schemes where the users may be divided in classes, such that, for example, 2 users of class 1 are enough to recover the secret, but if only one is available, the other may be replaced by any three users of class 2. These systems may be simulated with threshold schemes by giving more shares to members of more powerful classes, but this idea can be generalized without concerns as to how it is implemented.

Definition 2.10.3 (Generalized Secret Sharing) *Let w be a positive integer. $SS(\mathcal{K}, \mathcal{S}, d, r, \mathcal{Q})$ is a* secret sharing scheme *if $d : \mathcal{K} \rightarrow \mathcal{S}^w$ produces w shares of a key $k \in \mathcal{K}$ in such a way that only sets $Q \in \mathcal{Q}$ of users can compute the value of k by using the reconstruction function r, and no other group of participants can do so. The sets in $\mathcal{Q}$ are called qualified sets.*

2.10.2.3 Authentication Systems

An authentication system is a cryptographic construction that guarantees the integrity of messages. This means that if a user receives a message supposed to be from a given sender, he can be reasonably sure that that message was indeed sent by the indicated person and not forged by someone else, and also that the message was not altered in the way between the sender and the receiver.

Simmons developed a detailed theory of authentication systems in, for example, [Sim85], [Sim84] and [Sim88]. In [Sim88], he classifies authentication systems along three axis: systems with or without arbitration, systems with or without secrecy and systems computationally or provably secure.

In an authentication system, there are two legitimate users, which we call Alice and Bob. There is an opponent, Oscar, who can read, suppress and inject messages in the channel between Alice and Bob. Alice wants to send Bob one message, the source value, from a given set known to both users. As Simmons noted in [Sim85], the way that a system provides authentication is to create redundancy in the messages that

Alice can send Bob: for each source value that she may send to Bob, she can send several different messages, called coded or authenticated messages, but in a way that Bob will accept only a few of them as valid. As such, if Oscar tries to forge a message out of the blue and sends it to Bob, the latter will very likely reject this message. Bob and Alice share a secret key that allows Bob to sift through the redundancy in the system, so that Alice always sends a message that Bob accepts. This key is the encoding rule used by Alice to transform her source value in some coded message and then used by Bob to verify that the message he receives is within the set of valid messages.

Authentication systems can have secrecy. In this case, the authenticated message does not indicate the source value being communicated. Inversely, in an authentication system without secrecy, a coded message completely reveals the source value. For an example of why systems without secrecy are important, see [Sim88]. Authentication systems can also have splitting. In this case, each pair key/source-value may have several possible coded messages, and the encoding is done probabilistically. In system without splitting, there is only one value associated to each such pair. In this thesis, we only consider such systems.

2.10.2.4 Representation of Authentication Systems

An authentication system can be represented by a table that shows how the several encoding rules transform the source values into coded messages.

	0	**1**	**2**
0	0	3	6
1	1	4	7
2	2	5	8
3	0	5	7
4	1	3	8
5	2	4	6
6	0	4	8
7	1	5	6
8	2	3	7

Table 2.2: An example authentication system

The example in Table 2.2 corresponds to an example authentication system presented in [Sti02]. It is one of the smallest systems without secrecy with traditional uncon-

ditional security. The header line holds the source values, the leftmost column the secret-key, and the coded messages are the entries in the table. Notice that each coded message appears in only one column, that is, it determines the source value used to create it. As Stinson pointed in [Sti92], anyone knowing the system can create an isomorphic authentication system where each coded message is replaced by the source value and an authenticator. For ease of calculation, this authenticator can be defined as an integer enumerating the coded messages available for each source value. Thus, the previous code becomes the one in Table 2.3.

	S		
	0	**1**	**2**
0	(0,0)	(1,0)	(2,0)
1	(0,1)	(1,1)	(2,1)
2	(0,2)	(1,2)	(2,2)
3	(0,0)	(1,2)	(2,1)
4	(0,1)	(1,0)	(2,2)
5	(0,2)	(1,1)	(2,0)
6	(0,0)	(1,1)	(2,2)
7	(0,1)	(1,2)	(2,0)
8	(0,2)	(1,0)	(2,1)

Table 2.3: The same system, with the coded messages written as the concatenation of the source value and the authenticator

This table can be expanded into what is called an incidence matrix. We find it easier to explain the attacks against an authentication system with this kind of table than with a regular encoding matrix. In an incidence matrix, each column represents a different coded message and each row symbolizes a different key. A cell can be filled or not. In the first case, it means that there is some source value that the given key transforms into that particular coded message. The incidence matrix of our example system can be seen in Table 2.4. The incidence matrices of systems without splitting have exactly one 1 for each key in each group of columns under a specific value of $\mathcal{S}$, since each pair (k, s) can generate only one coded message. Also, every column has at least one 1 in it, otherwise the corresponding message could be excluded from the alphabet without altering the system.

We give an example system that will be used in later sections to clarify some points of the analysis of the security of individual instances. The system is insecure in the traditional sense and is defined by Table 2.5. Note that the sets of source values and

	S:	0			1			2		
	A:	0	1	2	0	1	2	0	1	2
	0	1	—	—	1	—	—	1	—	—
	1	—	1	—	—	1	—	—	1	—
	2	—	—	1	—	—	1	—	—	1
	3	1	—	—	—	—	1	—	1	—
K:	4	—	1	—	1	—	—	—	—	1
	5	—	—	1	—	1	—	1	—	—
	6	1	—	—	—	1	—	—	—	1
	7	—	1	—	—	—	1	1	—	—
	8	—	—	1	1	—	—	—	1	—

Table 2.4: The incidence matrix of the example system

keys is still the same, but that there is a difference in the several authenticators and consequently on the coded messages.

	S:	0			1		2			
	A:	0	1	2	0	1	0	1	2	3
	0	1	—	—	1	—	1	—	—	—
	1	1	—	—	1	—	—	—	1	—
	2	—	1	—	1	—	—	—	—	1
	3	—	1	—	1	—	—	—	1	—
K:	4	—	—	1	1	—	—	1	—	—
	5	—	—	1	1	—	—	1	—	—
	6	—	—	1	1	—	—	1	—	—
	7	1	—	—	—	1	1	—	—	—
	8	—	1	—	—	1	—	—	1	—

Table 2.5: An insecure system

In an impersonation attack, the attacker sends a forgery $(s, a) \in \mathcal{S} \times \mathcal{A}$ without seeing any valid message. The probability of the receiver accepting this message as valid is

$$\text{payoff}(s, a) = \sum_{k \in \mathcal{K},\, g_k(s,a)=1} \Pr[K = k].$$

In a substitution attack, the attacker knows that (s_1, a_1) is a valid message before sending a forgery $(s, a) \in \mathcal{S} \times \mathcal{A}$. The probability of the receiver accepting this

message as valid is

$$\text{payoff}(s, a, s_1, a_1) = \frac{\sum_{k \in \mathcal{K},\, g_k(s,a)=1,\, g_k(s_1,a_1)=1} \Pr[K = k]}{\sum_{k \in \mathcal{K},\, g_k(s_1,a_1)=1} \Pr[K = k]}.$$

Definition 2.10.4 *An authentication code without arbitration, without splitting and without secrecy is a tuple denoted* $AC(\mathcal{S}, \mathcal{A}, \mathcal{K}, f(k,s), g(k, \langle s, a \rangle), \alpha, \beta)$ *where* $\mathcal{S}$ *is the set of source states,* $\mathcal{A}$ *is the set of authenticators and* $\mathcal{K}$ *is the set of the secret keys. For each* $k \in \mathcal{K}$*, there is an injective encoding rule* $f_k : \mathcal{S} \mapsto \mathcal{A}$ *with* $f_k(s) = f(k,s)$ *that computes the message authentication code (mac) for each source value* $s \in \mathcal{S}$*. For each* $k \in \mathcal{K}$*, a verification function* $g_k : \mathcal{S} \times \mathcal{A} \mapsto \{0,1\}$ *with* $g_k(s,a) = g(k, \langle s, a \rangle)$ *can be defined as* $g_k(s,a) = 1$ *iff* $f(k,s) = a$*.*

The value α *is the maximum chance of success for an impersonation attack and* β *is the maximum chance of success for a substitution attack. Formally, for any fixed* $k \in \mathcal{K}$*,* $\max_{(s,a) \in \mathcal{S} \times \mathcal{A}} \text{payoff}(s,a) \leq \alpha$ *and* $\max_{(s,a),(s_1,a_1) \in \mathcal{S} \times \mathcal{A}} \text{payoff}(s, a, s_1, a_1) \leq \beta$*.*

2.10.2.5 Some Results About Authentication Systems

This section presents a few results about authentication systems. We give only those results needed in later proofs and do not try to cover the whole field of authentication systems without secrecy.

Definition 2.10.5 *Let* $\mathcal{K}_{s,a}$ *denote the set of keys that given* s *generate* a*, i.e., the keys authenticating a message* $\langle s, a \rangle$*. Equivalently, let* $m = \langle s, a \rangle$*. Then, define* $\mathcal{K}_m = \mathcal{K}_{s,a}$*.*

Definition 2.10.6 *Let* $\mathcal{A}_s$ *denote the set of authenticators that can be associated to some source value* s*.*

Theorem 2.10.7 *In an authentication system without secrecy, the total number of keys is equal to the number of authenticators that can be associated to each specific* s *times the average number of keys validating each coded message that can be associated with that* s*. Formally,* $|\mathcal{K}| = |\mathcal{A}_s| \cdot E_s(|\mathcal{K}_{s,a}|)$*, where we define* $E_s(|\mathcal{K}_{s,a}|) = \frac{\sum_{a \in \mathcal{A}_s} |\mathcal{K}_{s,a}|}{|\mathcal{A}_s|}$*.*

Corollary 2.10.8 *In an authentication system without secrecy, the number of keys is greater than or equal to the number of authenticators of a fixed source value* s *times the minimum number of keys validating a message that can be associated to that* s*:* $|\mathcal{K}| \geq |\mathcal{A}_s| \min_a \{|\mathcal{K}_{s,a}| : a \in \mathcal{A}_s\}$*.*

Corollary 2.10.9 *In an authentication system without secrecy, the number of keys is greater than or equal to the maximum number of authenticators times the minimum number of keys validating any message:* $|\mathcal{K}| \geq |\mathcal{A}| \cdot \min_{s,a}\{|\mathcal{K}_{s,a}| : a \in \mathcal{A}\}$.

Definition 2.10.10 *Let* $\mathcal{K}_{\langle s,a\rangle,\langle s',a'\rangle}$ *denote the set of keys that given* s *and* s' *generate respectively* a *and* a'. *Equivalently, let* $m = \langle s', a'\rangle$. *Then, define* $\mathcal{K}_{s,a,m} = \mathcal{K}_{\langle s,a\rangle,\langle s',a'\rangle}$

Definition 2.10.11 *Let* $\mathcal{A}_{s_0,s,a}$ *denote the set of authenticators generated from a source value* s_0 *by all the keys validating* $\langle s, a\rangle$. *In other words, it is the range of* $f(k, s_0)$ *where* $k \in |\mathcal{K}_{s,a}|$.

Theorem 2.10.12 *In an authentication system without secrecy, fix a pair* $\langle s, a\rangle$. *Then, for any* s_0, *the total number of keys authenticating* $\langle s, a\rangle$ *is equal to the number of authenticators generated from* s_0 *by the keys validating* $\langle s, a\rangle$ *times the average number of keys validating simultaneously* $\langle s, a\rangle$ *and* $\langle s_0, a_0\rangle$, *where* $\langle s_0, a_0\rangle$ *is a valid pair for the key in use:*$|\mathcal{K}_{s,a}| = |\mathcal{A}_{s_0,s,a}| \cdot E_{a_0}\{|\mathcal{K}_{s,a,s_0,a_0}| : a_0 \in \mathcal{A}_{s_0,s,a}\}$.

2.10.2.6 Commitment Systems

A Commitment Scheme is a cryptographic system whereby one party, which we call the "committer", wants to send a secret to another party, called the "verifier", in two steps, committing to her choice in a first instant and revealing that choice only at a second moment in time. They were introduced by Blum in his paper [Blu82] and are useful when both parties want to exchange secrets "simultaneously". A real world example is when a chess tournament match has to be adjourned, and one player seals her move without revealing it to the other party and without being able to change it when the game is resumed in a later session. A more serious application is when both parties need to generate a common string of perfect randomness for a task they're executing together without each of them being able to sway the string in her favour.

There are unconditionally and computationally secure commitment systems. Unconditionally secure commitment systems require a third participant, as defined by Rivest in [Riv99], and this is the only model we will analyse in this thesis.

Commitment schemes with a trusted initializer allow a sender to commit to a value and send that commitment to a receiver such that the value she committed to remains hidden from this. In a second step, the sender reveals her commitment and the receiver may verify that the sender is not fooling her. The third participant is required only

to give the other two some information that enables them to carry out the protocol. This third participant is completely honest and trusted by the other two. One might consider that a simpler solution would be, then, for the committer to send her message to the trusted party and let this send it to the receiver. However, this requires that all three participants be present at the moment the protocol is carried out. On the contrary, Rivest's model requires the presence of the third party only in a setup phase used to pre-distribute some information between the participants. The actual protocol is then carried out only between the other two parties.

A commitment scheme must satisfy a Concealing Property, i.e., the receiver can guess the value committed to only with a probability equal to a uniform random guess. On the other hand, the sender's commitment must effectively bind her, which means she can not open to the receiver a value different from her commitment. As shown in [BMSW02], a commitment system can not be completely binding, and so we say a system is $(1-\varepsilon)$-binding if the probability of the sender deceiving the receiver is at most ε.

The system has an alphabet $\mathcal{X}$ of secrets Alice can commit to, an alphabet $\mathcal{Y}$ of values that mask the secret and that Alice can send to Bob, an alphabet $\mathcal{K}$ of keys and an alphabet $\mathcal{V}$ of validation tags. There is a function f that for any key $k \in \mathcal{K}$ and any value $x \in \mathcal{X}$ produces a value $y = f(k,x) \in \mathcal{Y}$. For all k, the function $f_k(x) = f(k,x)$ is injective and defined for all $x \in \mathcal{X}$. All values in $\mathcal{Y}$ can be produced by at least some pair (k,x). There is also another function g that for any key $k \in K$ and validation tag $v \in \mathcal{V}$ produces a value in $\{0,1\}$, indicating whether the key k is accepted as valid by the tag v.

Definition 2.10.13 *A commitment scheme is a tuple denoted* $CM(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{V},f(k,x),g(v,k),\alpha,\beta)$ *where* $\mathcal{X}$ *is the source states alphabet,* $\mathcal{Y}$ *is the coded states alphabet,* $\mathcal{K}$ *is the alphabet of the committer's keys,* $\mathcal{V}$ *is the alphabet of the verifier's tags. For each* $k \in \mathcal{K}$*, there is an injective encoding rule* $f_k : \mathcal{X} \mapsto \mathcal{Y}$ *with* $f_k(x) = f(k,x)$ *that computes the encoding of each possible commitment* $x \in \mathcal{X}$*. For each* $v \in \mathcal{V}$*, there is a verification function* $g_v : \mathcal{K} \mapsto \{0,1\}$ *with* $g_v(k) = g(v,k)$*.*

The values α *and* β *are the maximum chances of success for the two kinds of attack described in Section 4.1.1. Formally,* $\max_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} \Pr[V=v] \leq \alpha$ *and* $\max_{k,k'\in\mathcal{K}} \frac{\sum_{v\in\mathcal{V}_{k'}\cap\mathcal{V}_k} \Pr[V=v]}{\sum_{v\in V_{k'}} \Pr[V=v]} \leq \beta$.

These schemes were mathematically formalized and studied in [BMSW02], where the authors prove some lower bounds on the binding probabilities and propose and analyse

implementations of optimally secure systems. They give a general description of a commitment scheme in Rivest's model that uses encoding and authentication keys and also a simplified scheme where the authentication key is not necessary. Then they offer a construction of commitment schemes based on resolvable designs and analyse its binding probabilities. They list two open problems: finding a lower bound on the amount of information that has to be pre-distributed to the users and sent by the sender to the receiver; and the existence of some relation between these schemes and authentication codes. The first of these questions was answered in [NMQO+03], while the second was answered in [PSMA07] and is the basis for Chapter 4 of this thesis.

Chapter 3

Individual Security of Cryptographic Systems

The aim of this chapter is to motivate the definition of notions of individual security of cryptographic systems. We analyse three kinds of systems for which there are proofs of unconditional security and show for each of them a definition of unconditional security based on the Kolmogorov complexity of individual instances. We show that if sufficiently many instances are secure according to this notion, then the system is secure under the traditional notion of information-theoretic security. Some of the results in this chapter appeared first in [Sal05] and [ALPS07].

Information theoretic security is based on the probabilities of guessing a given secret, be it the plaintext of a cipher system or the source-value of an authentication code, for example. A system is secure if the guessing probability of an attacker is not improved by knowledge of some public information, for example, a ciphertext or a set of private shares of a secret sharing system. These notions are usually applicable to random variables and provide a notion of security of the system as a whole, which on some cases means that the extra knowledge the attacker can have gives no information, in the information theoretic sense, about the secret she wants to find. On other occasions, we can only guarantee that the worst-case probability of success is not greater than some lower-bound for that probability. It bears noticing that in any cryptographic system there is always a positive chance that the attacker guesses the secret correctly. However, even though a system is information theoretically secure, it may have instances where the extra knowledge actually contains information about the secret. It is true the attacker has no way of knowing this information, otherwise the system wouldn't be secure, but we find that a secure system should not allow such

instances. It is very uncomfortable to have an encryption system produce a ciphertext that is exactly equal to the plaintext.

For that reason, we analyse the security of cryptographic systems at the level of individual instances, offering notions of security that are suitable for the instances of each system and study in what measure the security of instances influences the security of the system as a whole, as measured by traditional information theory. The notions of security we use are based on Kolmogorov Complexity, which provides a way to measure the intrinsic information of a string and the information it contains about another string, offering a parallel with the concepts of information based on Shannon's entropy function. We study three kinds of systems in this chapter: cipher systems, threshold secret sharing systems and authentication systems without secrecy.

3.1 Cipher Systems

We begin this study by examining the simplest cryptographic systems: cipher systems. In this section, we show what makes a cipher system secure both with entropy and Kolmogorov complexity and give an example system that is secure under both definitions: the one-time pad (OTP).

3.1.1 Motivation

Suppose Alice wants to send an enciphered text to Bob and does not want Eve to know what she's telling him. For maximum security, she uses the one-time pad. Ideally, the text Alice sends to Bob should hide the plaintext in such a way that the former does not reveal anything about the latter. But, what if Alice's random key turns out to be $0000111100001111\ldots$ or $010101010101\ldots$? Or even $0000\ldots$? It's easy to see the cipher texts will have a lot in common with the plain text. There is a simple rule describing how one can invert one to get the other. This is surely not what Alice wishes, for even if we exclude the extreme case where the cipher text is equal to the plaintext, these simple keys reveal too much about Alice's message.

Traditionally, Alice knows that even if this happens, Eve will be oblivious to it. Even if the message she holds has relevancy to the context where Alice and Eve are, it is only one of many relevant messages, and Eve can simply list all the texts of the given length and from these consider only those that are possible in their context... just as

she could without the ciphertext. However, Eve and Alice expect the key to be random and the ciphertext to look equally random, without any discernible pattern, since the probability of a cipher text being (almost) intelligible is so small. Then, if such a text appears, Eve may certainly be tipped off that something may have malfunctioned in the key generation. She may guess that Alice is using a key that is too simple, and that instead of the ciphertext being one possibility among all those that (almost) make sense, in this context or not, it is actually one of the possible ciphertexts that result from using a simple key. But simple keys are exponentially less than all keys, and Eve can quickly make a list of all possible plaintexts under simple keys for the ciphertext she holds. If Eve is correct in her assumptions, then the resulting set of plaintexts will not include all the texts of the required length, much less all of the texts that are possible in the actual context. If Eve is correct, then her uncertainty about Alice's message is greatly decreased and the system is effectively compromised. How then can Alice be sure that her key does not reveal information about the plaintext?

The answer is Kolmogorov complexity, and the associated notion of mutual information. We define the security of an instance by the amount of information the attacker needs to have to compute the plaintext and for that goal we use Kolmogorov complexity. This kind of analysis looks at security from a different prism than the usual one. The difference between the probabilities of guessing the plaintext with and without knowledge of the ciphertext is replaced by the difference between the amount of information necessary in each case. A simple key requires little information to be learned or recovered, and is therefore not recommended.

3.1.2 Information theoretic security of cipher systems

Recall that in the analysis of a cipher system we consider three participants: a sender, which we call Alice, a receiver, Bob, and a passive eavesdropper, Eve. Eve's objective is to find the secret message from Alice to Bob. She can try to simply guess the plaintext, where she will succeed with probability $\Pr[M = m]$ for the plaintext m. However, Eve can obtain the ciphertext that Alice sends Bob, and if the cipher system is bad, this might give her a large advantage. Her chances of guessing the plaintext, now, are $\Pr[M = m|C = c]$.

According to Shannon's definition, a cipher system is unconditionally secure if for any pair $(m, c) \in \mathcal{M} \times \mathcal{C}$ these two probabilities are equal. This implies that both distributions are independent and leads to the following definition of security.

Definition 3.1.1 *Consider a cipher system, and let M be a random variable over its plaintext messages and C a random variable over the respective ciphertexts. Then, the cipher system is unconditionally secure if and only if $H(M|C) = H(M)$.*

The essence of this definition is that the mutual information is 0. We can use $I(M;C)$ as the base of a more general notion of security, that tells us to what extent a system may be called secure, almost secure or not at all secure. The system is perfectly secure if $I(M;C) = 0$. When $I(M;C) > 0$, we can have an idea of the relative security of the system by computing how far the system is from the ideal.

Definition 3.1.2 *A private key cipher system has* δ relative security *if* $I(M;C) \leq \delta$.

This gives us a notion of security for the system as a whole. In the next section, we explore a notion of individual security.

3.1.3 Instance security for cipher systems

Let a plaintext $m \in \mathcal{M}$ and a key $k \in \mathcal{K}$ represent an instance of a cipher system, which we abbreviate by (m,k). Clearly, there's only one ciphertext associated to this instance. Intuitively, (m,k) is secure if the best *ad hoc* attack that can find the plaintext requires no more information than that given by this particular ciphertext and full information about the distribution of the messages, which is available to Eve under Kerckhoffs's principle. Of course, we cannot rule out the *ad hoc* attack where the attacker already knows the message and prints it out, but this requires full information about the plaintext.

Definition 3.1.3 *Let $(\mathcal{M}, \mathcal{C}, \mathcal{K}, e, d)$ be a cipher system, and μ be a distribution over $\mathcal{M} \times \mathcal{K}$. An instance $m, k \in \Sigma^n$ of the cipher system is* γ-secure, *or it has* γ secrecy, *if $I_K(m : e(k,m)|\mu) \leq \gamma$.*

This is very dependent on the "quality" of the key, since one can compute the plain text from the ciphertext if one can guess the key properly. Thus,

$$K(m|e(k,m)) \leq K(k) + O(1) \leq \log|\mathcal{K}| + O(1) = n + O(1).$$

We can see from this that simple keys yield ciphertexts that give much information about the plaintext, and that only incompressible keys, or Kolmogorov-random keys,

will afford proper security. This is not surprising, as a requirement for the information-theoretic security of this algorithm is that the key be sampled at random from all possible keys. It is therefore a recasting of an idea into a new model based on a different definition of randomness. We now prove that if enough instances of a cipher system are secure, then the system is nearly information theoretically secure. This result establishes that instance security is a sharper notion than information theoretic security.

Theorem 3.1.4 *For any private key cipher system $(\mathcal{M}, \mathcal{K}, \mathcal{C}, e, d)$, for any independent computable random variables M, K over $\mathcal{M}, \mathcal{K}$ with joint distribution μ, if the probability that an instance is γ secure is at least $(1-\varepsilon)$, then the system has $\gamma + \varepsilon \log(|\mathcal{M}|)$ secrecy, in the information theoretic sense.*

Furthermore, if for any t such that $\gamma < t$, $\mu(\{m, k : I_K(m : e(k,m)|\mu) = t\}) \leq f(t)$, then the system has $\gamma + \sqrt{\varepsilon \sum_{t>\gamma} t^2 f(t)}$ security.

Proof:

By Theorem 2.9.31 we have that up to an additive constant, $I(M;C) \leq \sum_{m,k} \mu(k,m) I_K(m : e(m,k)|\mu)$. We separate the sum into two parts, the secure instances and the others. Let G be the set of γ-secure instances.

$$\begin{aligned} I(M;C) &\leq \sum_{m,k \in G} \mu(k,m) I_K(m : e(m,k)|\mu) + \sum_{m,k \notin G} \mu(k,m) I_K(m : e(m,k)|\mu) \\ &\leq \gamma \sum_{m,k \in G} \mu(k,m) + \sum_{m,k \notin G} \mu(k,m)[K(m|\mu) - K(m|e(m,k),\mu)] \\ &\leq \gamma + \sum_{m,k \notin G} \mu(k,m) K(m|\mu) \\ &\leq \gamma + \varepsilon \log(|\mathcal{M}|). \end{aligned}$$

Considering the assumption in the last part of the theorem statement, $\mu(\{m, k : I_K(m : e(k,m)|\mu) = t\}) \leq f(t)$, then as before:

$$\begin{aligned} I(M;C) &\leq \gamma + \sum_{m,k \notin G} \mu(k,m) I_K(m : e(m,k)|\mu) \\ &= \gamma + \sum_{t>\gamma} t \cdot \mu(\{m, k : I_K(m : e(k,m)|\mu) = t\}) \\ &\leq \gamma + \sum_{t>\gamma} t\sqrt{f(t)} \cdot \sqrt{\mu(\{m, k : I_K(m : e(k,m)|\mu) = t\})}. \end{aligned}$$

Recall that the Cauchy-Schwarz inequality (Theorem 2.3.3) states that $\sum_i a_i b_i \leq \sqrt{(\sum_i a_i^2)(\sum_i b_i^2)}$ and this implies that

$$\begin{aligned} I(M;C) &\leq \gamma + \sqrt{\sum_{t>\gamma} t^2 f(t)} \sqrt{\sum_{t>\gamma} \mu(\{m,k : I_K(m : e(k,m)|\mu) = t\})} \\ &\leq \gamma + \sqrt{\sum_{t>\gamma} t^2 \cdot f(t)} \sqrt{\varepsilon}. \end{aligned}$$

□

3.1.4 Instance security of one-time pad

In the previous section, we gave a definition of security of individual instances of cipher systems, and shown that if sufficiently many instances are secure, then the system is secure in the traditional sense, thereby establishing that our definition is a refinement of the standard notion of security. In this section, we consider a specific cipher system: the one-time pad. First (Theorem 3.1.5), we identify a set of individually secure instances. These are the keys with maximum Kolmogorov complexity, together with messages that have no information about the key. These conditions are a direct counterpart of the independence of the key and the message, and the randomness of the key, in the traditional analysis. Then, in (Corollary 3.1.6), we show that this set is indeed large enough to imply almost perfect security in the information theoretic sense.

Theorem 3.1.5 *Let μ be a distribution over $\Sigma^n \times \Sigma^n$. Let $m, k \in \Sigma^n$ be an instance of the one-time pad, i.e., $e(k,m) = m \oplus k$, with $K(k|\mu) \geq n - \alpha$, and $K(m,k|\mu) \geq K(m|\mu) + K(k|\mu) - \beta$ where $\oplus$ represents the bitwise exclusive-or of the arguments. Then the instance has $\alpha + \beta$ secrecy.*

Proof: By Theorem 2.9.17, up to an additive constant,

$$\begin{aligned}
K(m|m \oplus k, \mu) &\geq K(m|m \oplus k, K(m \oplus k|\mu), \mu) \\
&= K(m|\mu) - K(m \oplus k|\mu) + K(m \oplus k|m, K(m|\mu), \mu) \\
&= K(m|\mu) - K(m \oplus k|\mu) + K(k|m, K(m|\mu), \mu) \\
&\geq K(m|\mu) - n + K(k|m, K(m|\mu), \mu) \quad \text{(using } |m \oplus k| = n\text{)} \\
&\geq K(m|\mu) - n + K(k, m|\mu) - K(m|\mu) \\
&\geq K(m|\mu) - n + n - \alpha - \beta \quad \text{(by hypothesis)} \\
&= K(m|\mu) - \alpha - \beta.
\end{aligned}$$

□

Combining this with Theorem 3.1.4, we have the following corollary.

Corollary 3.1.6 *Let μ_M be a computable distribution over $\mathcal{M}$, and μ_K be the uniform distribution. Then one-time pad is $O(1)$-secure in the information theoretic sense.*

Proof: Let $\mu(m, k) = \mu_M(m) \cdot \mu_K(k)$. Let

$$G_{\alpha,\beta} = \{(m, k) : K(k|\mu) \geq n - \alpha \text{ and } K(m, k|\mu) \geq K(m|\mu) + K(k|\mu) - \beta\}.$$

By Theorem 3.1.5, we know that all instances in $G_{\alpha,\beta}$ have $\alpha + \beta$ secrecy. We show that $\mu((\mathcal{M}\times\mathcal{K}) \setminus G_{\alpha,\beta}) \leq (2^{-\alpha} + 2^{-\beta}))$. Observe that $\mathcal{M} \times \mathcal{K} \setminus G_{\alpha,\beta} \subseteq B_\alpha \cup B_\beta$, where

$$\begin{aligned}
B_\alpha &= \{(m, k) : K(k|\mu) < n - \alpha\}, \\
B_\beta &= \{(m, k) : K(m, k|\mu) < K(m|\mu) + K(k|\mu) - \beta\}.
\end{aligned}$$

By Theorem 2.9.10, $\mu(B_\alpha) \leq 2^{-\alpha}$.

Claim 1 $\mu(B_\beta) \leq 2^{-\beta}$.

By definition, $\mu(B_\beta) = \sum_{m,k\in B_\beta} \mu(m, k) = \sum_m \mu_M(m) \sum_{k:m,k\in B_\beta} \mu_K(k)$. By Theorem 2.9.17, $K(m, k|\mu) = K(m|\mu)+K(k|m, K(m|\mu), \mu)$, so when m is fixed, the second summation runs over k with $K(k|m, K(m|\mu), \mu) < K(k|\mu) - \beta$. By Theorem 2.9.10, there are at most $2^{K(k|\mu)-\beta}$ terms in the inner summation. Therefore, using $\mu_K(k) = 2^{-n}$ for all k and consequently $K(k|\mu) \leq n$, $\mu(B_\beta) \leq \sum_m \mu_M(m)2^{K(k|\mu)-\beta}\mu_K(k) \leq 2^{-\beta}$, which concludes the proof of the claim.

Claim 2 *For any t, $\mu(\{m, k : I_K(m : e(k, m)|\mu) = t\}) \leq 2^{-t}$.*

The claim follows from the fact that

$$\begin{aligned} I_K(m : e(k,m)|\mu) &= K(e(k,m)|\mu) - K(e(k,m)|m, K(m|\mu), \mu) \\ &\leq n - K(e(k,m)|m, K(m|\mu), \mu) \\ &= n - K(k|m, K(m|\mu), \mu) \end{aligned}$$

where the first equality comes from symmetry of information and the final equality holds in the case of the one-time pad. Therefore $\mu(\{m,k : I_K(m : e(k,m)|\mu) = t\}) \leq \mu(\{m,k : K(k|m, K(m|\mu), \mu) \leq n-t\}) \leq 2^{n-t} \cdot 2^{-n} = 2^{-t}$. We can apply Theorem 3.1.4 with $\gamma = \alpha + \beta$ and $\varepsilon = 2^{-\alpha} + 2^{-\beta}$. Therefore, the system has $\gamma + \sqrt{\varepsilon \sum_{t>\gamma} t^2 2^{-t}}$ secrecy. By Lemma 2.3.2, $\sum_{t \geq \gamma+1} t^2 2^{-t} = \frac{(\gamma+2)^2+2}{2^\gamma}$. This is maximum when $\gamma = 0$, so the system has security at most $\alpha + \beta + \sqrt{6\varepsilon}$ security. If $\alpha = \beta = O(1)$, then we can conclude that the one-time pad is $O(1)$-secure. □

3.1.5 Non-uniform distribution of keys

We can also analyse the case where the keys are sampled from a distribution that is not uniform, but instead has min-entropy at least $n - \delta$.

Theorem 3.1.7 *Let μ_M be a computable distribution over $\mathcal{M}$, and μ_K be a distribution on keys with min-entropy at least $n - \delta$. Then, one-time pad is $O(2^\delta)$-secure in the information theoretic sense.*

Proof: The proof is the same as that of Corollary 3.1.6, except that now we have to take into account the different distribution μ_K. We only list the differing points, assuming the whole rest of the proof. The differences come from the fact that now the probability of each key being sampled is at most $2^{-(n-\delta)}$, instead of being exactly 2^{-n}.

Claim 3 $\mu(B_\alpha) \leq 2^{\delta-\alpha}$.

Proof: By Theorem 2.9.10, there are at most $2^{n-\alpha}$ strings in B_α, and since all of them have probability at most $2^{-(n-\delta)}$, we get $\mu(B_\alpha) \leq 2^{n-(n-\delta)-\alpha} = 2^{\delta-\alpha}$. □

Claim 4 $\mu(B_\beta) \leq 2^{\delta-\beta}$.

Proof: Just as before, the number of strings in B_β is at most $2^{K(k|\mu)-\beta} \leq 2^{n-\beta}$. Then, $\mu(B_\beta) \leq 2^{n-\beta-(n-\delta)} = 2^{\delta-\beta}$. □

Claim 5 *For any t, $\mu(\{m,k : I_K(m : e(k,m)|\mu) = t\}) \leq 2^{\delta - t}$.*

Proof: This also follows from the same reasoning as before. We get $\mu(\{m,k : I_K(m : e(k,m)|\mu) = t\}) \leq \mu(\{m,k : K(k|m, K(m|\mu), \mu) \leq n - t\}) \leq 2^{n-t} \cdot 2^{-(n-\delta)} = 2^{\delta - t}$. □

Now, we apply Theorem 3.1.4. We still let $\gamma = \alpha + \beta$ and $\varepsilon = 2^{-\alpha} + 2^{-\beta}$ and determine that the system has $\gamma + \sqrt{2^{\delta} \cdot \varepsilon \sum_{t>\gamma} t^2 2^{\delta - t}}$ secrecy. This can be simplified to $\gamma + 2^{\delta}\sqrt{\varepsilon \sum_{t>\gamma} t^2 2^{-t}} \leq \alpha + \beta + 2^{\delta}\sqrt{6\varepsilon}$. As before, making $\alpha = \beta = O(1)$, we conclude the one-time pad is $O(2^{\delta})$-secure for key distributions with min-entropy at least $n - \delta$. □

3.1.6 Resource-bounded instance security of one-time pad

In this section, we prove that if one is willing to expend the time necessary to produce a secure instance, then it is guaranteed to be secure against an adversary that is limited in the amount of time at its disposal to decrypt the instance.

Definition 3.1.8 *Let $(\mathcal{M}, \mathcal{C}, \mathcal{K}, e, d)$ be a cipher system. An instance (m,k) of the cipher system is* γ-secure against a t-time-bounded adversary *if $K^t(m|e(k,m)) \geq K^t(m) - \gamma$.*

Theorem 3.1.9 *For any polynomial time bound t, there is a time bound t' polynomial in t such that the following holds: if $m, k \in \Sigma^n$ and $e(k,m) = m \oplus k$ are an instance of a one-time pad scheme, such that $CAM^{t'}(k,m) \geq n + K^t(m) - \alpha$, then the instance has $\log^{O(1)} n + \alpha$ secrecy against a t-time-bounded adversary.*

Proof: The proof is similar to Theorem 3.1.5, except for the application of time bounded symmetry of information, Theorem 2.9.29.

$$\begin{aligned} K^t(m|m \oplus k, \mu) &\geq K^t(m, (m \oplus k, \mu)) - K^t(m \oplus k, \mu) \\ &\geq CAM^{t'}(m) + CAM^{t'}(m \oplus k, \mu|m) - K^t(m \oplus k, \mu) - O(\log^3(|x| + |y|)) \\ &\geq CAM^{2t'}(m,k) - n - O(\log^3(|x| + |y|)) \\ &\geq K^t(m) - \alpha - O(\log^3(|x| + |y|)). \end{aligned}$$

□

To compute the CAM complexity of a key and message pair, one could simulate all CAM programs up to length $2n$ (an exponential number) to rule out the existence of a short program. This is by no means efficient; however, it is computable.

3.1.7 Weak keys

The one-time pad is perhaps not the ideal system for the previous analysis, for an attacker can not ever know if she has got the right key and therefore if she has all the information needed to get the corresponding plaintext. Even a brute force attack is useless for the attacker. But things become a little different if we suppose that the attacker can effectively recognize when she has cracked the secret, that is, when she has access to an oracle that answers whether its input is the true plaintext in question. This might happen if, for example, the system was used to protect a password or a safe combination that can not be changed after it was initially set. It is only a matter of time until an attacker exhausts all possibilities and opens the box, but the time required is exponential in the length of the password. Then, the box itself functions as the oracle described above. In such a scenario, one possible attack for a time-bounded adversary is to run through low-complexity keys, say, those strings of length n with complexity not greater than $c \log n$. If the actual secret key is weak, having complexity at most $c \log n$ for some constant c, then a short program will reveal it and the secret will be cracked in $O(n^c)$ time.

There are other cryptosystems where, however, such oracles exist. For example, there are known weak keys in systems like DES or IDEA and it is still a common discussion whether the key-generation algorithms should test whether a generated key is weak. The implicit assumption in such argument is that an attacker can actually test all the weak keys before launching a more determined attack, since this first step can usually be done quickly and thus does not have a real impact in the complexity of the breaking attempt. Weak keys form a very small set of a supposedly secure cipher system and so their complexity has to be low. They form a set of keys that share a certain characteristic. But there may be different sets of weak keys for a cipher system, sporting different properties that are discovered over time. That means that the sets of weak keys known at a certain moment for a given system might not be exhaustive, and that even if a key-generator checks for all the weak keys it knows, it might actually be allowing some weak-key not yet discovered to be so. However, such a weak key would surely have low Kolmogorov complexity, as said above, which means that if the key-generator instead checks all keys with low time-bounded Kolmogorov complexity then it is effectively ruling out all weak keys, even those that are still not known as such.

3.2 Threshold Secret Sharing Schemes

This section is dedicated to the study of threshold secret sharing schemes, applying the same principles outlined in the previous sections.

3.2.1 Motivation

Just like the case of a cipher system, in a secret sharing system there is a piece of information that must be kept secret from one or more users who hold information related to that secret. The basic requirements of security are therefore the same, and accordingly we use in this section the same philosophy we used for cipher systems in what regards the definition of individual secure instances.

Before proceeding, we give here a few definitions to simplify the notation in this section.

Definition 3.2.1 *Let* $[w] = \{1, 2, \ldots, w\}$ *represent the set of all participants. Let* $\mathbf{xy_w} = \langle (x_1, y_1), \ldots, (x_w, y_w) \rangle$ *represent the concatenation of all shares in the same ordering as used for the participants.*

For any $\mathcal{B} \subseteq [w]$, *let* $\mathbf{xy}_{\mathcal{B}}^{j} = \mathbf{xy_w} \restriction \mathcal{B}$ *be the projection of* $\mathbf{xy_w}$ *onto the participants selected by* $\mathcal{B}$, *that is,* $\mathbf{xy}_{\mathcal{B}}^{j} = \langle (x_{i_1}, y_{i_1}), \ldots, (x_{i_j}, y_{i_j}) \rangle$ *where* $\mathcal{B} = \{i_1, i_2, \ldots, i_j\}$ *for some* $j \leq w$. *For any instance* $(k, \mathbf{xy_w})$ *and* $\mathcal{B} \subseteq [w]$, $(k, \mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}})$ *is a* $|\mathcal{B}|-$*arrangement of the instance* $(k, \mathbf{xy_w})$. *We also write* $\mathbf{xy_{q-1}}$ *to represent any concatenation of some* $q-1$ *distinct pairs of* $\mathbf{xy_w}$, *without considering which subset of* $[w]$ *originated it.*

We need a shorthand definition for the remainder of this section.

Definition 3.2.2 *Fix an integer* w *and another integer* $q \leq w$. *Let the notation* $\binom{[w]}{q-1}$ *represent the set of subsets of* $[w]$ *that have* $q-1$ *elements. Formally,* $\binom{[w]}{q-1} = \{\mathcal{B} : \mathcal{B} \subseteq [w], |\mathcal{B}| = q-1\}$. *Context will dictate whether this notation is to be read as this set or as its cardinality.*

3.2.2 Information theoretic security of threshold secret sharing schemes

Let $\mathcal{P} = \{P_i, 1 \leq i \leq w\}$ denote the set of participants of a threshold secret sharing system, and let $D \notin \mathcal{P}$ be the dealer, that is, a special participant who chooses the

value of the secret key k.

Let $\mathcal{B} = (i_1, \cdots, i_j)$ be any set of participants that want to reconstruct the key, where $j \leq w$. Let $d(k) = \{(x_i, y_i) : 1 \leq i \leq w\}$ be the set of all shares distributed by the dealer, where the values x_i are public and each y_i is known only by its holder. The values y_i are determined from the secret information held by D, which includes at least the secret k. Let X be a random variable over the possible values for x_i, and Y be analogous for y_i. Then, Y is totally dependent of X, that is, $H(X,Y) = H(X)$. In what follows, let K be a random variable over the possible values for k and $X^j_{\mathcal{B}}$ be a random variable over the public values of the users in $\mathcal{B}$, and analogously for $Y^j_{\mathcal{B}}$ and the private shares, where $j = |\mathcal{B}|$.

Definition 3.2.3 *A (q, w) threshold scheme $(\mathcal{K}, \mathcal{S}, d, r)$ has δ* security *if for a set of attackers $\mathcal{B}$ we have that:*

- *If $|\mathcal{B}| \geq q$ then $H(K|X^j_{\mathcal{B}}, Y^j_{\mathcal{B}}) = 0$;*
- *If $|\mathcal{B}| < q$ then $H(K|X^j_{\mathcal{B}}, Y^j_{\mathcal{B}}) \geq H(K) - \delta$.*

3.2.3 Individual secrecy of secret sharing schemes

We now give an individual analysis of perfect security based on Kolmogorov complexity.

Contrary to the case of the cipher system, it is not adequate in this case to define an instance to be simply the particular occurrence of the secret value and all the information available to a particular attacker. This is so because in this scheme we must consider simultaneously the attacks of all groups of users with less than q members, not just one. As such, each instance of a threshold secret sharing system must be described by the public and private shares of all the users of the system. The secret could also be considered part of the instance, but it is completely determined by the complete set of shares, so it is not required. The following definition takes all this into account, and defines security for an individual instance of such systems.

Definition 3.2.4 *Let $(\mathcal{K}, \mathcal{S}, d, r)$ be a (q, w) threshold scheme and μ a distribution over $\mathcal{K} \times \mathcal{S}^w$, and $(k, \mathbf{xy_w})$ an instance of this scheme.*

1. *$(k, \mathbf{xy}^{\mathbf{q-1}}_{\mathcal{B}})$ is $\gamma-$secure against an attack from $\mathcal{B}$ if $I_K(k : \mathbf{xy}^{\mathbf{q-1}}_{\mathcal{B}}|\mu) \leq \gamma$.*

2. $(k, \mathbf{xy_w})$ *is* (γ, ϕ)*-secure if* $\Pr_{\mathcal{B}\in\binom{[w]}{q-1}}[I_K(k : \mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu) \leq \gamma] \geq 1 - \phi$ *under the uniform distribution over* $q-1$ *arrangements.*

The first point addresses the base case of security against an isolated attack of a single set. We have chosen to model security for the maximal sets of users that still must not be able to recover the secret, as any smaller set will necessarily have at most as much information as these sets. In short, the arrangement resists such an attack if the information that the public and private shares of the colluded users contain about the secret is less than a certain parameter.

The second point gives the notion of security for the system viewed globally. It basically says that the probability of finding some group that has too much information about the secret, over all maximal groups of illegal size, is very small. Using this definition, we proceed as in the section dedicated to cipher systems, by giving first a top-level proof that enough secure instances imply good information-theoretic security.

Theorem 3.2.5 *For any* (q, w)*-threshold scheme where* $\mathcal{K}$ *is the set of keys and* $\mathcal{S} = \{(x_i, y_i) : 1 \leq i \leq w\}$ *the set of all shares, for any variables* K, S^w *over* $\mathcal{K}, \mathcal{S}^w$ *with distribution* $\mu(\mathbf{xy_w})$ *over the total shares of the users, where the several public shares are i.i.d., if the probability that any given instance is* (γ, ϕ)*-secure is at least* $(1 - \varepsilon)$*, then the system has* $(\gamma + (\varepsilon + \phi)\log|\mathcal{K}|)$ *secrecy, in the information theoretic sense.*

Proof:

1. By definition of (q, w)-threshold secret sharing scheme, if $|\mathcal{B}| \geq q$ and all elements of $\mathcal{B}$ have different shares, the attackers can effectively compute the secret merely by pooling their private and public shares. This means there is a function that computes K from any set of q distinct points $\langle x_i, y_i\rangle$, and therefore $H(K|\mathbf{xy_w}\restriction\mathcal{B}) = 0$.

2. Let $\mathbf{xy_w} = \langle(x_1, y_1), \ldots, (x_w, y_w)\rangle$ represent a particular legal instance of the system. Since this includes all the shares, $\mathbf{xy_w}$ uniquely determines the secret and we can consider the complete instance $(k, \mathbf{xy_w})$.

   We show that $I(K; X^{q-1}Y^{q-1})$ is bounded above by $\gamma + (\varepsilon + \phi)\log|\mathcal{K}|$, using the upper bound given by Theorem 2.9.31, as the average of Kolmogorov mutual information. We first consider fixed instances of the scheme, then take the average over all instances. For any instance $(k, \mathbf{xy_w})$, let

$$\tilde{I}(k : \mathbf{xy_w}|\mu) = E_{\mathcal{B}\sim U\binom{[w]}{q-1}}(I_K(k : \mathbf{xy_w}\restriction\mathcal{B}\ |\mu))$$

represent the average mutual information that an unqualified group of $q-1$ users can gain about the secret. Let μ' be derived from μ as the marginal distribution of μ, with k being uniquely determined from $\mathbf{xy_w}$. Let G be the set of instances that are (γ, ϕ) secure. Then, for all $(k, \mathbf{xy_w}) \in G$,

$$\begin{aligned}
\tilde{I}(k : \mathbf{xy_w}|\mu) &= \sum_{\mathcal{B}: I_K(k:\mathbf{xy_w}\restriction\mathcal{B})\leq\gamma} \mu'(k, \mathbf{xy_w}\restriction\mathcal{B}|\mathbf{xy_w}) I_K(k : \mathbf{xy_w}\restriction\mathcal{B}|\mu) \\
&\quad + \sum_{\mathcal{B}: I_K(k:\mathbf{xy_w}\restriction\mathcal{B})>\gamma} \mu'(k, \mathbf{xy_w}\restriction\mathcal{B}|\mathbf{xy_w}) I_K(k : \mathbf{xy_w}\restriction\mathcal{B}|\mu) \\
&\leq \gamma + \Pr[\mathcal{B} : I_K(k : \mathbf{xy_w}\restriction\mathcal{B}) > \gamma] \cdot K(k|\mu) \\
&\leq \gamma + \phi \cdot \log|\mathcal{K}|.
\end{aligned}$$

The average value of $\tilde{I}(k : \mathbf{xy_w}|\mu)$ over all instances is

$$\begin{aligned}
E_{\mathbf{xy_w}\sim\mu}\tilde{I}(k : \mathbf{xy_w}|\mu) &= E_{\mathbf{xy_w}\sim\mu} E_{\mathcal{B}\sim U\binom{[w]}{q-1}} I_K(k : \mathbf{xy_w}\restriction\mathcal{B}\ |\mu) \\
&= E_{k,\mathbf{xy_{q-1}}\sim\mu'} I_K(k : \mathbf{xy_{q-1}}\ |\mu).
\end{aligned}$$

By Theorem 2.9.31,

$$\begin{aligned}
I(K : X^{q-1}, Y^{q-1}) &\leq E_{k,\mathbf{xy_{q-1}}\sim\mu'} I_K(k : \mathbf{xy_{q-1}}\ |\mu) \\
&= E_{\mathbf{xy_w}\sim\mu}\tilde{I}(k : \mathbf{xy_w}|\mu) \\
&= E_{\mathbf{xy_w}\sim\mu}\tilde{I}(k : \mathbf{xy_w}|\mu, \mathbf{xy_w} \in G) \\
&\quad + E_{\mathbf{xy_w}\sim\mu}\tilde{I}(k : \mathbf{xy_w}|\mu, \mathbf{xy_w} \notin G) \\
&\leq \gamma + \phi \cdot \log|\mathcal{K}| + \varepsilon \cdot \log|\mathcal{K}| \\
&= \gamma + (\phi + \varepsilon)\log|\mathcal{K}|.
\end{aligned}$$

□

3.2.4 Instance security of Shamir's scheme

As we did for the one-time pad, we now study a particular implementation of a secret sharing system. The Shamir (q, w)-threshold scheme (see [Sha79]) in $\mathbb{Z}_p$, with $p \geq w + 1$, has two phases:

Initialization phase D publicly chooses w non-zero elements of $\mathbb{Z}_p$, denoted by x_i, $1 \leq i \leq w$. For $1 \leq i \leq w$, D gives the value x_i to P_i.

Share distribution Suppose D wants to share a key $k \in \mathbb{Z}_p$. D secretly chooses independently at random $q-1$ elements of $\mathbb{Z}_p$, $a_1, \ldots, a_{q-1}$ and constructs a random polynomial $a(x) = K + \sum_{j=1}^{q-1} a_j x^j \bmod p$, where $a(x) \in \mathbb{Z}_p[x]$ of degree at most $q-1$. Then, for $1 \leq i \leq w$, D computes the secret share $y_i = a(x_i)$ and gives it to participant P_i.

For $1 \leq i \leq w$, every participant P_i obtains a point (x_i, y_i) on this polynomial where all the coefficients $a_0, \ldots, a_{q-1}$ are unknown elements of $\mathbb{Z}_p$ and $a_0 = k$ is the key. A set of users $\mathcal{B} \subseteq \mathcal{P}, \mathcal{B} = \{P_{i_j}, 1 \leq i_j \leq w, 1 \leq j \leq q\}$ can reconstruct the key by means of polynomial interpolation like the Lagrange formula, which is an explicit formula to recover $a(x)$ given q points $(x_{i_1}, y_{i_1}), \ldots, (x_{i_q}, y_{i_q})$ on the polynomial. We identify the secure instances of Shamir's scheme as the ones where the key and the shares are independent according to Kolmogorov complexity.

Lemma 3.2.6 *Let $(k, \mathbf{xy_w})$ be an instance of the Shamir secret sharing scheme. For any set $\mathcal{B} \subseteq [w]$ with $|\mathcal{B}| = q-1$, if $K(k, \mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu) \geq K(k|\mu) + |\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}| - \alpha$ then $I_K(k : \mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu) \leq \alpha + O(1)$.*

Proof: By assumption,

$$K(k, \mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu) \geq K(k|\mu) + |\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}| - \alpha,$$

and symmetry of information brings

$$\begin{aligned} K(k, \mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu) &= K(\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu) + K(k|\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}, K(\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu), \mu) \\ &\leq |\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}| + K(k|\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}, K(\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu), \mu) + O(1). \end{aligned}$$

Therefore,

$$\begin{aligned} & K(k|\mu) + |\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}| - \alpha - O(1) \leq K(k|\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}, K(\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu), \mu) + |\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}| \\ \Leftrightarrow \quad & K(k|\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}, K(\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu), \mu) \geq K(k|\mu) - \alpha - O(1) \\ \Leftrightarrow \quad & I_K(k : \mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu) \leq \alpha + O(1). \end{aligned}$$

□

For Shamir's scheme, we can infer information theoretic security from instance security, as follows.

Theorem 3.2.7 *Under the uniform distribution over the random shares and a computable distribution over the secret, Shamir's secret sharing scheme is* $(\log\log p + 1)$ *secure in the information theoretic sense.*

Proof: For a given instance $(k, \mathbf{xy_w})$, and any $\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}$ arrangement derived from $\mathbf{xy_w}$, $K(k, \mathbf{xy_w}) \leq K(k, \mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}) + |\mathbf{x_{w-q+1}}|$ where $\mathbf{x_{w-q+1}} = \langle x_1, x_2, \ldots, x_w \rangle \restriction ([w] \setminus \mathcal{B})$, that is, the public shares of all users not present in $\mathcal{B}$. This is because if we are given the description of $q-1$ private shares and the secret k we can recover the secret polynomial and from there, using the remaining public values, compute all the missing private shares. Then,

$$K(\mathbf{xy_w}|\mu) = K(k, \mathbf{xy_w}|\mu) \leq \min_{\mathcal{B}\subseteq[w], |\mathcal{B}|=q-1} K(k, \mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu) + |\mathbf{x_{w-q+1}}|.$$

This means that if $K(k, \mathbf{xy_w}|\mu) \geq K(k|\mu) + |\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}| + |\mathbf{x_{w-q+1}}| - \alpha$, then for all arrangements $\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}$ of $q-1$ users made from $(k, \mathbf{xy_w})$

$$\begin{aligned} K(k|\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}, \mu) &\geq K(k|\mu) + |\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}| + |\mathbf{x_{w-q+1}}| - \alpha - |\mathbf{x_{w-q+1}}| \Leftrightarrow \\ K(k|\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}, \mu) &\geq K(k|\mu) + |\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}| - \alpha \end{aligned}$$

which implies, by Theorem 3.2.6, $I_K(k : \mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}|\mu) \leq \alpha + O(1)$. Then, by definition, this instance will be $(\alpha, 0)$ secure.

Using $K(k, \mathbf{xy_w}|\mu) = K(\mathbf{xy_w}|\mu)$, the probability that $(k, \mathbf{xy_w})$ satisfies the above condition is:

$$\Pr_{(\mathbf{xy_w})}[K(\mathbf{xy_w}|\mu) \geq K(k|\mu) + |\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}| + |\mathbf{x_{w-q+1}}| - \alpha] = \sum_{\mathbf{xy_w}\in\mathcal{W}} \mathcal{U}_w(\mathbf{xy_w})$$

for $\mathcal{W} = \{\mathbf{xy_w} : K(\mathbf{xy_w}|\mu) \geq K(k|\mu) + |\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}| + |\mathbf{x_{w-q+1}}| - \alpha\}$.

In Shamir's scheme, the key and the public shares are all drawn independently from the same alphabet $\mathcal{K}$. Also, by construction, any q distinct private shares are also independent. So, a whole instance may be coded by a string with just $(w+q)\log p$ bits. Then, $|\mathbf{xy}_{\mathcal{B}}^{\mathbf{q-1}}| = 2(q-1)\log p$, $|\mathbf{x_{w-q+1}}| = (w-q+1)\log p$ and $|\mathbf{xy_w}| = (w+q)\log p$.

Now we apply Theorem 2.9.10 to find there is at least a fraction $1 - \frac{2^{(1+2(q-1)+(w-q+1))\log p - \alpha}}{2^{(q+w)\log p}} = 1 - \frac{1}{2^\alpha}$ of secure instances, where we used the fact that $K(k|\mu) \leq \log p$. Finally, let $\alpha = \log\log p$ and apply Theorem 3.2.5 with $\varepsilon = \frac{1}{p\log p}$, $\gamma = \log\log p$ and $\phi = 0$. Then, the system has security $\log\log p + 1$. □

3.3 Unconditional Security of Authentication Codes Without Secrecy

This section addresses systems that have a different kind of security from the ones previously studied. In an authentication system, the attacker does not want to find a secret, she wants to forge an illegal message. The requirement of security can no longer be expressed as the value of mutual information between two strings or random variables. Instead, security is measured by the chances of an attacker being successful.

In this section, we make a brief summary of the main results in the literature about unconditionally secure authentication systems without splitting nor secrecy. Then, we propose an individual notion of security, as was done in previous sections for other systems, and analyse the relation between this and the classical notion.

Simmons described the two basic attacks, impersonation and substitution, and showed that there are no authentication systems that can prevent forging with absolute certainty. To do this, he analysed the possibilities of success of each attack and gave lower bounds for them that are always positive. One of these bounds was simultaneously and independently published by Brickell ([Bri84]). These bounds were functions of the entropies of the random variables representing the system. Stinson, in [Sti87] and [Sti92], gave combinatorial lower bounds for the same probabilities, for systems without splitting, both with and without secrecy. He also gave constructions for systems of both kinds.

These attacks were extended in the literature to cases where Oscar sees some legitimate messages before he launches his attack. Massey ([Mas86]) defines a spoofing attack of order i as Oscar's attempt to forge a new message after having seen i legitimate messages from Alice. This notion generalizes the basic impersonation and substitution attacks which are, respectively, a spoofing attack of order 0 and a spoofing attack of order 1. Constructions and bounds for these generalized attacks are given, for example, in [dS90] and [Sti88]. Maurer ([Mau00]) has a slightly different terminology, considering that in the generalized impersonation Oscar sees $i-1$ messages sent by Alice and lets them go through to Bob and then he forges an i^{th} message and hopes Bob accepts it; in the generalized substitution, Bob sees i messages sent by Alice but lets only the first $i-1$ reach Bob. He then alters the i^{th} message and sends this modified one to Bob. Usually, Oscar is not worried in forging a message that will make Bob accept a specific source value: Oscar wins if the source value Bob sees is different from the one Alice sent. Maurer, in [Mau00], was the first to analyse attacks that are

successful only if Oscar makes Bob accept a specific source value or if he learns the value Bob uncovers.

3.3.1 Information theoretic security

In this thesis, we are only interested in the basic impersonation and substitution attacks against an authentication system without secrecy, splitting or arbitration. Following, we give a summary of the bounds in the literature for the impersonation and the substitution attacks. We use the notation P_{d_i} to represent the probability of success for a spoofing attack of order i. Therefore, P_{d_0} represents the probability of an impersonation attack and P_{d_1} that of a substitution attack.

3.3.1.1 Impersonation Attack

The probability that Bob accepts $(s, a) \in \mathcal{S} \times \mathcal{A}$ as authentic is

$$\text{payoff}(s, a) = \Pr_K[a = e_k(s)] = \sum_{k \in \mathcal{K}_{s,a}} \Pr[K = k].$$

To maximize his chance of success, Oscar chooses the (s, a) that maximizes his payoff(s, a). Hence, $P_{d_0} = \max\{\text{payoff}(s, a) : s \in \mathcal{S}, a \in \mathcal{A}\}$.

Lemma 3.3.1 *Let $m = \langle s, a \rangle$ be an instance of an authentication system without secrecy. Then,* payoff$(s, a) = \Pr[\mathcal{A} = a | \mathcal{S} = s]$.

Proof: Let $K_{s,a} = \{k \in K : e_k(s) = a\}$. We can write $\Pr[M = m] = \Pr[S = s, A = a] = \Pr[S = s] \cdot \Pr[A = a | S = s]$ and by definition of payoff, we have that

$$\begin{aligned} \Pr[M = m] &= \Pr[S = s] \cdot \sum_{k \in K_{s,a}} \Pr[K = k] \\ &= \Pr[S = s] \cdot \text{payoff}(s, a). \end{aligned}$$

This implies payoff$(s, a) = \Pr[A = a | S = s]$. □

Simmons has shown that in any authentication system this probability is always positive:

Theorem 3.3.2 *([Sim85])*

For any authentication code $(\mathcal{S}, \mathcal{A}, \mathcal{K}, e)$ without splitting where the key is chosen independently from the source value, $P_{d_0} \geq 2^{-H(K)+H(K|A,S)} = 2^{-H(A|S)}$.

Proof: We prove this in two steps. First the relation to $H(A|S)$. Let $p(s,a)$ and $p(a|s)$ be simplified notations of $\Pr[A = a, S = s]$ and $\Pr[A = a|S = s]$. Then, $P_{d_0} = \max_{s,a}[p(a|s)] \Leftrightarrow -\log P_{d_0} = \min_{s,a}[-\log p(a|s)]$. This implies that $-\log P_{d_0} \leq E_{s,a}[-\log p(a|s)] = -\sum_{s,a} p(s,a) \log p(a|s) = H(A|S)$ which in turn brings $\log P_{d_0} \geq -H(S|A) \Leftrightarrow P_{d_0} \geq 2^{-H(S|A)}$. Finally, we prove the equality.

$$\begin{aligned}
-H(K) + H(K|S,A) &= H(K|S,A) - H(K) \\
&= H(K,A,S) - H(S,A) - H(K) \\
&= H(K) + H(A,S|K) - H(S,A) - H(K) \\
&= H(S|K) + H(A|S,K) - H(S) - H(A|S) \\
&= H(S) - H(S) + 0 - H(A|S) \\
&= -H(A|S).
\end{aligned}$$

The fourth equality follows because of independence of the source-value and key, and the fifth follows because knowing the source value and the key the authenticator can be unambiguously computed. □

Stinson ([Sti88, Sti92]) has given combinatorial bounds for this attack and we list them next.

Definition 3.3.3 *Denote by* $\Pr[K_{s,a}]$ *the sum of the probabilities of the keys that associate the source-value* s *to authenticator* a*, that is,* $\Pr[K_{s,a}] = \sum_{k \in \mathcal{K}_{s,a}} \Pr[K = k]$. *As seen before,* $\Pr[K_{s,a}] = \text{payoff}(s,a)$.

Theorem 3.3.4 *In an authentication code* $(\mathcal{S}, \mathcal{A}, \mathcal{K}, e)$ *without splitting,* $P_{d_0} \geq \frac{|\mathcal{S}|}{|\mathcal{M}|}$.

Proof: Fix some source-state s. Then, $\sum_{a \in \mathcal{A}_s} \Pr[K_{s,a}] = 1$. Clearly, $\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_s} \Pr[K_{s,a}] = |\mathcal{S}|$. Dividing by $|\mathcal{M}|$, we get the average probability of the payoff of an impersonation attack. By an averaging argument, there must be a message with payoff greater than or equal to this average, therefore $P_{d_0} \geq |\mathcal{S}|/|\mathcal{M}|$. □

Other bounds can be given that correspond closely to Simmons lower bounds. We give a counterpart to $\log P_{d_0} \geq H(K|S,A) - H(K)$:

Theorem 3.3.5 *For an authentication code* $(\mathcal{S}, \mathcal{A}, \mathcal{K}, e)$ *without splitting* $P_{d_0} \geq \frac{E_{s,a}(|K_{s,a}|)}{|K|}$.

Proof: From Theorem 3.3.4,

$$\begin{aligned} P_{d_0} \geq \frac{|\mathcal{S}|}{|\mathcal{M}|} &= \frac{|\mathcal{S}||\mathcal{K}|}{|\mathcal{M}||\mathcal{K}|} \\ &= \frac{\sum_{s,a} |\mathcal{K}_{s,a}|}{|\mathcal{M}||\mathcal{K}|} \\ &= \frac{E_{s,a}|\mathcal{K}_{s,a}|}{|\mathcal{K}|}. \end{aligned}$$

□

Another theorem, present in [Sti92], shows that the determinant factor for the security against an impersonation attack is the number of available authenticators. Stinson has shown $P_{d_0} \geq 1/|\mathcal{A}|$. We give other versions for when the number of authenticators for each source value is not constant. All these variants are counterpart to the bound $\log P_{d_0} \geq H(A|S)$:

Theorem 3.3.6 *For an authentication code $(\mathcal{S}, \mathcal{A}, \mathcal{K}, e)$ without splitting, $P_{d_0} \geq \frac{1}{E_s|\mathcal{A}_s|}$.*

Proof: Note that $|\mathcal{M}| = \sum_s |\mathcal{A}_s|$. From Theorem 3.3.4, $P_{d_0} \geq \frac{|\mathcal{S}|}{|\mathcal{M}|} = \frac{|\mathcal{S}|}{\sum_s |\mathcal{A}_s|} = \frac{1}{E_s(|\mathcal{A}_s|)}$. □

Corollary 3.3.7 *For an authentication code $(\mathcal{S}, \mathcal{A}, \mathcal{K}, e)$ without splitting, $P_{d_0} \geq \frac{1}{\min_s |\mathcal{A}_s|}$.*

Proof: Since $\min_s |\mathcal{A}_s| \leq E_s|\mathcal{A}_s|$, then $P_{d_0} \geq \frac{1}{E_s(|\mathcal{A}_s|)} \geq \frac{1}{\min_s |\mathcal{A}_s|}$. □

3.3.1.2 Substitution Attack

Now we examine the substitution attack. There is a perfect parallel between this and the impersonation attack, coming from the fact that in both of them the objective is the same, and the only changing factor is the knowledge of a valid message.

Fix a message $m = \langle s_1, a_1 \rangle$ sent from Alice to Bob. Then, the chance of success of a substitution attack for a message $\langle s, a \rangle$ is

$$\begin{aligned} \text{payoff}(s, a, m) &= \frac{\sum_{k \in K_{s,a,m}} \Pr[K = k]}{\sum_{k \in K_m} \Pr[K = k]} \\ &= \Pr_K(a = e_k(s) | a_1 = e_k(s_1)). \end{aligned}$$

By a similar reasoning to that of Lemma 3.3.1, payoff$(s, a, m) = \Pr[A = a|S = s, M = m]$. As before, we define $P_{d_1} = \max\{\text{payoff}(s, a, m) : s \in \mathcal{S}, a \in \mathcal{A}, m \in \mathcal{M}\}$.

Brickell was the first to give a lower bound for the probability of success of this attack:

Theorem 3.3.8 *([Bri84]) For any authentication code $(\mathcal{S}, \mathcal{A}, \mathcal{K}, e)$ without splitting $P_{d_1} \geq 2^{-H(A|S,M)} \geq 2^{-H(K|M)}$.*

We give a proof similar to that of Theorem 3.3.2.

Proof: Let $p(s, a|m)$ and $p(a|s, m)$ be simplified notations of $\Pr[A = a, S = s]$ and $\Pr[A = a|S = s]$. Then,

$$\begin{aligned}
P_{d_1} &= \max_{s,a,m}[p(a|s,m)] \Leftrightarrow \\
-\log P_{d_1} &= \min_{s,a,m}[-\log p(a|s,m)] \Rightarrow \\
-\log P_{d_1} &\leq E_{s,a,m}[-\log p(a|s,m)] \\
&= -\sum_{s,a,m} p(s,a,m)\log p(a|s,m) = H(A|S,M) \Rightarrow \\
\log P_{d_1} &\geq -H(S|A,M) \Leftrightarrow \\
P_{d_1} &\geq 2^{-H(S|A,M)}.
\end{aligned}$$

Next, we prove the inequality $H(K|M) \geq H(A|S, M)$. We develop $H(K, A, S|M)$ in two ways. First,

$$\begin{aligned}
H(K,A,S|M) &= H(K|M) + H(A,S|K,M) \\
&= H(K|M) + H(S|K,M) + H(A|S,K,M) \\
&= H(K|M) + H(S|M)
\end{aligned}$$

where the last line follows from independence of S and K given M and the fact that knowing the source value and the key the authenticator is completely defined. Secondly,

$$\begin{aligned}
H(K,A,S|M) &= H(A,S|M) + H(K|A,S,M) \\
&= H(A|S,M) + H(S|M) + H(K|A,S,M).
\end{aligned}$$

Gathering everything, we have

$$\begin{aligned}
&H(A|S,M) + H(K|A,S,M) = H(K|M) \Rightarrow \\
&H(A|S,M) \leq H(K|M) \Rightarrow \\
&P_{d_1} \geq 2^{-H(A|S,M)} \geq 2^{-H(K|M)}.
\end{aligned}$$

□

An analysis similar to the combinatorial bounds of Section 3.3.1.1 can be made for the impersonation attack. We merely list the following result, which offers a parallel to Theorem 3.3.5.

Theorem 3.3.9 *For an authentication code $(\mathcal{S}, \mathcal{A}, \mathcal{K}, e)$ without splitting where the keys are uniformly distributed, and the number of keys validating each message is constant and equal to $|K_m|$, $P_{d_1} \geq \frac{E_{s,a,m}|K_{s,a,m}|}{|K_m|}$.*

Proof: By definition, $\text{payoff}(s,a,m) = \frac{\sum_{k \in \mathcal{K}_{s,a,m}} \Pr[K=k]}{\sum_{k \in \mathcal{K}_m}} = \frac{|\mathcal{K}_{s,a,m}|/|\mathcal{K}|}{|\mathcal{K}_m|/|\mathcal{K}|} = |\mathcal{K}_{s,a,m}|/|\mathcal{K}_m|$.

Since $P_{d_1} = \max_{s,a,m} \text{payoff}(s,a,m)$ it must happen that $P_{d_1} \geq \frac{E_{s,a,m}|\mathcal{K}_{s,a,m}|}{|\mathcal{K}_m|}$. □

Definition 3.3.10 *From the previous lower bounds, an authentication code is said to have perfect security if those bounds are attained, i.e., $P_{d_0} = 2^{-H(A|S)}$ and $P_{d_1} = 2^{-H(A|S,M)}$.*

We generalize the notion of security in the following definition:

Definition 3.3.11 *An authentication code $(\mathcal{S}, \mathcal{A}, \mathcal{K}, e)$ has (δ, δ') security if*

1. $\log P_{d_0} \leq -H(A|S) + \delta$;
2. $\log P_{d_1} \leq -H(A|S,M) + \delta'$.

3.3.2 Individual Security Measures

We now focus strictly on the security of individual instances. The bounds of Section 3.3.1 have shown that the fundamental quantity for security is the probability of guessing a good authenticator for some source state s. Using this and the insight of previous sections, we let the complexity of computing an authenticator be the main measure of security against an attack.

3.3.2.1 Impersonation Attack

Definition 3.3.12 *The difficulty of an impersonation attack for a source value s is the amount of information needed for computing the right authenticator for it: $K(a|s)$.*

To turn a difficulty measure into a security measure, it must be compared to a fixed security threshold. Instances with difficulty very near the threshold are considered secure, whereas other instances are insecure. To do this, we need to find a value that can be intuitively seen as an appropriate threshold for $K(a|s)$. An authenticator a can be described by its index in the set A_s to which it belongs. Thus,

$$K(a|s) \leq K(A_s|s) + \log |A_s|.$$

Since Oscar knows the encoding table of the whole system, he can easily compute A_s from s so $K(A_s|s) = O(1)$. Thus, the maximum value of $K(a|s)$ is at most $\log |A_s|$. However, if $|A_s|$ is very low, for example 1 in the extreme case, then the instance is surely not secure, as the chance of success is at least $1/|A_s|$. So we can not use $|A_s|$ as the comparison value.

From the bounds in Section 3.3.1, we can view the quantity $|M|/|S|$ as a measure of the redundancy introduced in the system by the designer and consider it as the cost paid to guarantee a lower bound on the success probability equal to $|S|/|M|$. This has been seen to be equal to the average number of authenticators per source value so by an averaging argument, the maximum number of authenticators is at least as big as $|M|/|S|$. Each A_s is a subset of the largest A_s, i.e. A, and therefore the maximum $|A_s|$ is $|A|$. This value has several advantages as the security threshold:

- Since $\log |A|$ is the maximum of all $\log |A_s|$, no $K(a|s)$ can be greater than that, except up to a small additive constant. Thus, the difference $\log |A_s| - K(a|s)$ is always positive, again up to a fixed additive constant.
- From the reasoning in Section 3.3.1.1, for a fixed source value the less authenticators exist the highest will the lower bound of the success probability be. Inversely, the largest $|A|$ the more can the designer of the system expect to reduce the probabilities of success. The system can still be badly designed enough to not take advantage of it, but the potential is there anyway.

Example 1 *Consider again the example system of Table 2.5. The maximum number of authenticators is* 4*, giving the potential to design a system with a lower bound on the success probability of* $1/4$*. However, the authenticators are mostly wasted because two of the three source messages do not use all of them. We can be sure that the probability of success for source value* 0 *is at least* $1/3$ *and for source value* 1 *it is at least* $1/2$*. By the way, the more general lower bound* $|M|/|S|$ *is* $1/3$*. But if the system has to provide enough bits to distinguish between* 4 *different authenticators, it could have used them*

all for every source value without increasing the length of the coded messages. Thus, optimal systems will have the number of authenticators equal to maximum for all s.

3.3.2.2 Substitution Attack

The analysis of the substitution attack is similar to the previous one. Following the same reasoning as before, it can be said that the difficulty of a successful substitution attack is the difficulty of finding the authenticator a for a chosen source value s such that Bob will accept the message $\langle s, a\rangle$ as valid given that a valid message $m = \langle s_1, a_1\rangle$ is known.

Definition 3.3.13 *The difficulty of a substitution attack for a source value s, given that a valid message m was sent by Alice to Bob, is the difficulty of computing the right authenticator for it: $K(a|s, m)$.*

As done before, we compare this measure with a suitable threshold. Recall the example system given above (Table 3.1). There are two coloured columns under

	S:	0			1		2			
	A:	0	1	2	0	1	0	1	2	3
	0	1	—	—	1	—	1	—	—	—
	1	1	—	—	1	—	—	—	1	—
	2	—	1	—	1	—	—	—	—	1
	3	—	1	—	1	—	—	—	1	—
K:	4	—	—	1	1	—	—	1	—	—
	5	—	—	1	1	—	—	1	—	—
	6	—	—	1	1	—	—	1	—	—
	7	1	—	—	—	1	1	—	—	—
	8	—	1	—	—	1	—	—	1	—

Table 3.1: An insecure system

source value 0, corresponding to two possible authenticators. The keys that generate these authenticators are highlighted with the same colour, and so are the messages for different source values that those keys validate. Suppose Alice sends Bob message $(0, 2)$, the second coloured column. Then Oscar knows Bob has key 4, 5 or 6. But looking at the columns for source values 1 or 2, he also sees that whatever the key Bob has, the right authenticator for the source value chosen is always the same,

this is, he has absolute certainty of fooling Bob. If Alice had sent message $(0,0)$ instead, corresponding to the first coloured column, Oscar would have a harder time deciding what message to send, but still far from ideal. For source value 1, Bob could accept either message, depending on whether he has key 7 or one of the other two. Oscar would probably send message $(1,0)$ since this is validated by two possible keys instead of one. But for source value 2 Oscar's chances are better. Out of the four possible messages, Bob will certainly reject two. Oscar only has to choose between the remaining two. The crucial point here is that the message sent by Alice and the source value picked by Oscar determine a set of possible authenticators that make valid messages. Call this set $A_{s,m}$. And just as in the impersonation case the security hinged on the number of elements of A_s, now it depends on the number of elements in $A_{s,m}$. And when $|A_{s,m}| < |A_s|$, the particular instance (s,m) is not secure.

As such, we can immediately see that in the previous example there is no security if Alice sends any message with source value 0, because each of them is validated by three keys only which are insufficient to cover the four authenticators for source value 2. In this scenario, Oscar will be able to exclude always at least one possible coded message. Message $(0,2)$ is absolutely insecure, because for $s = 1,2$, it happens that $|A_{s,(0,2)}| = 1$ giving Oscar total certainty in his attack. In this respect, message $(1,0)$ would be the most secure against a substitution attack for it wouldn't allow Oscar to exclude any possible authenticator for either of the other source values. However, it is still extremely insecure against an impersonation attack. As was done for the impersonation case, the security threshold we use is the maximum possible value of $|\mathcal{A}_{s,m}|$, that is, $|\mathcal{A}|$.

3.3.2.3 Instance Security

The previous discussion for the basic attacks allows us to give the following definition. For technical reasons, we have altered the quantities discussed in Section 3.3.2 to include in the conditional knowledge of the probability distribution that rules the system. We can assume this distribution is known under Kerckhoffs's principle.

Definition 3.3.14 *Let $(\mathcal{S}, \mathcal{A}, \mathcal{K}, e)$ be an authentication code without splitting and μ a distribution over $\mathcal{S} \times \mathcal{K}$.*

- *an instance (s,a) is γ secure against impersonation if $K(a|s, K(s|\mu), \mu) \geq \log(|\mathcal{A}|) - \gamma$.*

- *an instance* (s, a, m) *is* γ' *secure against substitution if* $K(a|s, m, K(s, m|\mu), \mu) \geq \log(|\mathcal{A}|) - \gamma'$.

Note that $K(a|s, m) \leq \log |\mathcal{A}_{s,m}|$ and so $\log |\mathcal{A}_{s,m}|$ must be at least $\log |\mathcal{A}| - \gamma'$ for an instance (s, k, m) to be secure. The following theorem allows us to relate the deception probabilities to Kolmogorov based security of instances.

Theorem 3.3.15 *For any authentication code* $(\mathcal{S}, \mathcal{A}, \mathcal{K}, e)$ *without splitting, for any independent variables* S, K *over* $\mathcal{S}, \mathcal{K}$ *with computable distribution* μ *and any large enough polynomial time bound* t,

1. *for any instance* (s, a) *of the authentication code,*

$$2^{-\max_{s,a} K^t(a|s,K(s|\mu),\mu)} \leq P_{d_0} \leq 2^{-\min_{s,a} K^t(a|s,K(s|\mu),\mu)}.$$

2. *for any instance* (s, a, m) *of the authentication code,*

$$2^{-\max_{a,s,m} K^t(a|s,m,K(s,m|\mu),\mu)} \leq P_{d_1} \leq 2^{-\min_{a,s,m} K^t(a|s,m,K(s,m|\mu),\mu)}.$$

Proof: For the first statement,

$$\begin{aligned}
\max_{s,a} K^t(a|s, K(s|\mu), \mu) &\geq \sum_{s,a} p(s, a) K^t(a|s, K(s|\mu), \mu) \\
&\geq H(A|S) \\
&\geq \log(1/P_{d_0}) \\
&\geq \min_{s,a} K^t(a|s, K(s|\mu), \mu).
\end{aligned}$$

The first inequality is trivial; the second follows from Theorem 2.9.30. The third is Theorem 3.3.2. Finally, the last inequality follows by using the Shannon-Fano code, which encodes a source $\{(x, p(x))\}$ with codewords of length $\log 1/p(x)$. This encoding can be carried out in polynomial time if the distribution μ is given.

Similarly, for the second statement (using Theorem 3.3.8 instead of Theorem 3.3.2)

$$\begin{aligned}
\max_{a,s,m} K^t(a|s, m, K(s, m|\mu), \mu) &\geq \sum_{a,s,m} p(a, s, m) K^t(a|s, m, K(s, m|\mu), \mu) \\
&\geq H(A|S, M) \\
&\geq \log(1/P_{d_1}) \\
&\geq \min_{a,s,m} K^t(a|s, m, K(s, m|\mu), \mu).
\end{aligned}$$

□

Corollary 3.3.16 *For any authentication code $(\mathcal{S}, \mathcal{A}, \mathcal{K}, e)$ without splitting,*

1. *if all instances (s, a) are γ-secure against an impersonation attack and*
2. *if all instances (s, a, m) are γ'-secure against a substitution attack,*

then the authentication code is (γ, γ')-secure in the information theoretic sense.

3.3.3 Secure Systems

Even though Lemma 3.3.16 only says that an authentication system is unconditionally secure if all, and not most of, its instances are unconditionally secure, there is still some interest in identifying the secure instances of a system known to be unconditionally secure according to traditional analysis. We show that given some reasonable assumptions about an instance, we can say it is secure in the individual sense. This theorem does not identify all secure instances, but gives sufficient conditions for an instance to be secure.

Theorem 3.3.17 *For an instance (k, m) of an authentication system without secrecy, without splitting and without arbitration, where $m = \langle s, a \rangle$, if the key k and s are independent, the key is Kolmogorov random and the number of keys authenticating m is not much greater than the minimum number of keys authenticating any m, then the instance is secure against an impersonation attack.*

Formally, if $K(k, s) \geq K(k) + K(s) - O(1)$ and $K(k) \geq |k| - O(1)$ and if $|K_{s,a}| \leq O(1) \cdot \min\{|K_{s,a}| : \langle s, a \rangle \in M\}$, then $K(a|s) \geq \log |A| - O(1)$.

Proof: By Lemma 3.3.19, $K(a|s^*) \geq I_K(k : m) - O(1)$. Then,

$$\begin{aligned} K(a|s^*) &\geq K(k) - K(k|m^*) - O(1) \\ &\geq K(k) - \log |K_{s,a}| - O(1) \qquad (3.1) \\ &\geq |k| - \log |K_{s,a}| - O(1) \\ &\geq \log |K| - \log |K_{s,a}| - O(1) \qquad (3.2) \end{aligned}$$

where (3.1) follows from $K(k|\langle s, a \rangle^*) \leq K(K_{s,a}|\langle s, a \rangle) + \log |K_{s,a}| \leq \log |K_{s,a}| + O(1)$. Also (3.2) follows from the fact that if a set has $\log |K|$ elements, then at least $\log |K|$ bits are needed to describe them (we tacitly assume all elements in a set are of the

same size). Now, by Corollary 2.10.9, $\log|K| \geq \log|A| + \log\min\{|K_{s,a}|, \langle s,a\rangle \in M\}$ and so (3.2) is greater than or equal to

$$\log|A| + \log\min\{|K_{s,a}|, \langle s,a\rangle \in M\} - \log|K_{s,a}| - O(1) \geq$$
$$\log|A| - O(1)$$

where the last line follows from the assumption

$$|K_{s,a}| \leq O(1) \cdot \min\{|K_{s,a}| : \langle s,a\rangle \in M\}.$$

□

We now show that some reasonable assumptions about an instance imply that it is secure against a substitution attack.

Theorem 3.3.18 *For an instance $(k, m = \langle s,a\rangle, m_1 = \langle s_1, a_1\rangle)$ of an authentication system without secrecy, let k be the key, m_1 be the valid message known to Oscar and m be the message that Oscar wants Bob to accept as valid. If,*

$$K(k, s_1) \geq K(k) + K(s_1) - O(1), \tag{3.3}$$
$$K(k, s|m_1^*) \geq K(k|m_1^*) + K(s|m_1^*) - O(1), \tag{3.4}$$
$$K(k) \geq \log|K| - O(1), \tag{3.5}$$
$$|K_{s_1,a_1}| \leq O(E_{a_i}(|K_{s_1,a_i}|, a_i \in A_{s_1})), \tag{3.6}$$
$$|K_{s,a,s_1,a_1}| \leq O(E_{a_i}(|K_{s,a_i,s_1,a_1}|, a_i \in A_{s,s_1,a_1})), \tag{3.7}$$
$$|A| \leq O(|A_{s,s_1,a_1}|), \tag{3.8}$$

then $K(a|s, m_1) \geq \log|A| - O(1)$.

In plain English, the previous conditions mean that the key and the source value of the known valid message are unconditionally independent (3.3)*; the key and the source value of the attacking message are independent in the presence of the known valid message* (3.4)*; the key is maximally random* (3.5)*; the number of keys validating $\langle s_1, a_1\rangle$ is not much more than the average number of keys validating messages for source value s_1* (3.6)*; the number of keys validating both $\langle s,a\rangle$ and $\langle s_1, a_1\rangle$ is not much more than the average number of keys validating $\langle s_1, a_1\rangle$ and any other message for source value s* (3.7) *and the number of authenticators possible for $\langle s_1, a_1\rangle$ and s is not far from the maximum* (3.8)*.*

Before we give the proof, we present the following claim.

Claim 6 *In an authentication system without secrecy, if* $K(k, s_1) \geq K(k) + K(s_1) - O(1)$, *then we have* $K(a_1|s_1^*) \geq K(k) - K(k|m_1^*) - O(1)$.

Proof: Take the alternative representation of $K(m_1|k^*)$, $K(s_1, a_1|k^*)$. By application of the addition theorem and the fact that the authenticator can be computed from the key and the source value, we get $K(m_1|k^*) = K(s_1|k^*)$ up to an additive constant.

By assumption, we have $K(m_1|k^*) = K(s_1|k^*) \geq K(s_1) - O(1)$. This brings

$$\begin{aligned} K(k) &= K(k, m_1) - K(m_1|k^*) \\ &\leq K(k, m_1) - K(s_1) + O(1) \\ &= K(k|m_1^*) + K(m_1) - K(s_1) + O(1) \\ &= K(k|m_1^*) + K(s_1, a_1) - K(s_1) + O(1) \\ &= K(k|m_1^*) + K(a_1|s_1^*) + O(1). \end{aligned}$$

□

Proof: (Of Theorem 3.3.18) By assumption and Claim 6, $K(k|m_1^*) + K(a_1|s_1^*) \geq K(k) \geq \log|K| - O(1)$. By Theorem 2.10.7, $\log|K| = \log|A_{s_1}|E_{a_i}(|K_{s_1,a_i}|, a_i \in A_{s_1})$ and by assumption this brings $\log|K| \geq \log|A_{s_1}| + \log|K_{s_1,a_1}| - O(1)$. Plugging this above, brings

$$\begin{aligned} &K(k|m_1^*) + K(a_1|s_1^*) \geq \log|A_{s_1}| + \log|K_{s_1,a_1}| - O(1) \Leftrightarrow \\ &K(k|m_1^*) \geq \log|K_{s_1,a_1}| - O(1) \end{aligned} \tag{3.9}$$

where the last line follows from $K(a_1|s_1^*) \leq \log|A_{s_1}| + O(1)$.

By assumption (3.4) and Lemma 3.3.20, we can write

$$\begin{aligned} &K(a|\langle s, m_1\rangle^*) \geq I_K(k : m|m_1^*) - O(1) = \\ &K(k|m_1^*) - K(k|\langle m, m_1\rangle^*) \geq \\ &K(k|m_1^*) - \log|K_{m,m_1}| - O(1), \end{aligned}$$

and joining with (3.9),

$$K(a|\langle s, m_1\rangle^*) \geq \log|K_{s_1,a_1}| - \log|K_{m,m1}| - O(1). \tag{3.10}$$

By Corollary 2.10.9 and assumption (3.7)

$$\log|K_{s_1,a_1}| \geq \log|A_{s,s_1,a_1}| + \log|K_{s,a,s_1,a_1}| - O(1). \tag{3.11}$$

Joining (3.10) and (3.11) we get

$$\begin{aligned}
&K(a|\langle k, s_1\rangle^*) \geq \log|A_{s,s_1,a_1}| + \log|K_{s,a,s_1,a_1}| - \log|K_{s,a,s_1,a_1}| - O(1) \Leftrightarrow \\
&K(a|\langle k, s_1\rangle^*) \geq \log|A_{s,s_1,a_1}| - O(1) \Rightarrow \\
&K(a|\langle k, s_1\rangle^*) \geq \log|A| - O(1)
\end{aligned}$$

where the last line follows from assumption (3.8). □

To exemplify these theorems, we present an unconditionally secure authentication system given by Stinson in [Sti02]. Fix some prime p. Then, let $\mathcal{K} = \{0, 1, \ldots, p^2 - 1\}, \mathcal{S} = \mathcal{A} = \{0, 1, \ldots, p-1\}$. Interpret k as a vector (i, j) where $j = k \bmod p$ and $i = (k - j)/p$. Define $e_k(s) = is + j \bmod p$. If the keys are uniformly distributed, this system is unconditionally secure. Every s has the same number of authenticators, every message is validated by the same number of keys and any two messages corresponding to different source values are validated by the same number of keys. Algebraically, this is:

$$\begin{aligned}
&|\mathcal{A}_s| = |\mathcal{A}| = p, \ \forall s \in \mathcal{S}; \\
&|\mathcal{K}_m| = p, \quad \forall m \in \mathcal{M}; \\
&|\mathcal{K}_{m,m_1}| = 1, \ \forall m = \langle s, a\rangle, m_1 = \langle s_1, a_1\rangle \in \mathcal{M}, s \neq s_1.
\end{aligned}$$

Such a scheme easily satisfies assumptions (3.6) to (3.8) of Theorem 3.3.18 for all instances. The remaining assumptions are natural and occur with high probability, as can be shown following similar reasonings to those used for the one-time pad.

3.3.3.1 Auxiliary Proofs

Lemma 3.3.19 *If a key k and a source value s of an authentication system without secrecy are independent, that is, $K(k, s) \geq K(k) + K(s) - O(1)$, then up to an additive constant $K(a|s^*) \geq I_K(k : m) - O(1)$, where k is a key that authenticates $m = \langle s, a\rangle$.*

Proof: We can develop $K(k, m)$ in two ways:

$$\begin{aligned}
K(k, m) &= K(k, a, s) \\
&= K(k, s) + K(a|\langle k, s\rangle^*) \\
&= K(k, s) + O(1) \\
&\geq K(k) + K(s) - O(1)
\end{aligned}$$

since a is completely determined by k and s.

$$\begin{aligned} K(k,m) &= K(m) + K(k|m^*) \\ &= K(s,a) + K(k|m^*) \\ &= K(s) + K(a|s^*) + K(k|m^*). \end{aligned}$$

Joining both we have

$$\begin{aligned} K(k) - O(1) \leq K(a|s^*) + K(k|m^*) - O(1) \Leftrightarrow \\ K(k) - K(k|m^*) - O(1) \leq K(a|s^*) - O(1) \Rightarrow \\ I_K(k:m) - O(1) \leq K(a|s^*). \end{aligned}$$

□

Lemma 3.3.20 *For an authentication system without secrecy where m_1 is a valid message and k is the secret key held by Alice and Bob and if the source value and the key are still independent after a valid message is known, then up to an additive constant:*

$$K(a|\langle s, m_1\rangle^*) \geq I_K(k:m|m_1^*) - O(1).$$

Formally, we state the assumption as $K(k,s|m_1^*) \geq K(k|m_1^*) + K(s|m_1^*) - O(1)$.

Proof: Similarly to what was done in Lemma 3.3.19, we develop $K(m,k,m_1)$ in two different ways:

$$\begin{aligned} K(m,k,m_1) &= K(a,s,k,m_1) \\ &= K(s,k,m_1) + K(a|\langle s,k,m_1\rangle^*) \\ &= K(s,k,m_1) \\ &= K(k|\langle s,m_1\rangle^*) + K(s,m_1) \end{aligned}$$

since the authenticator a is completely determined from the source value and the key. Also,

$$\begin{aligned} K(m,k,m_1) &= K(m,m_1) + K(k|\langle m,m_1\rangle^*) \\ &= K(a,s,m_1) + K(k|\langle m,m_1\rangle^*) \\ &= K(a|\langle s,m_1\rangle^*) + K(s,m_1) + K(k|\langle m,m_1\rangle^*). \end{aligned}$$

Joining both derivations, we get

$$K(k|\langle s,m_1\rangle^*) = K(a|\langle s,m_1\rangle^*) + K(k|\langle m,m_1\rangle^*). \tag{3.12}$$

By Lemma 2.9.19, $K(k,s|m_1^*) - K(s|m_1^*) = K(k|\langle s, m_1\rangle^*)$ and coupled with the assumption of the theorem, this gives

$$K(k|\langle s, m_1\rangle^*) \geq K(k|m_1^*) - O(1).$$

Continuing from (3.12),

$$\begin{aligned}
K(k|m_1^*) - O(1) \leq K(a|\langle s, m_1\rangle^*) + K(k|\langle m, m_1\rangle^*) \Leftrightarrow \\
K(a|\langle s, m_1\rangle^*) &\geq K(k|m_1^*) - K(k|\langle m, m_1\rangle^*) - O(1) \\
&= K(k|m_1^*) - K(k, m, m_1) + K(m, m_1) - O(1) \\
&= K(k|m_1^*) - (K(k, m|m_1^*) + K(m_1)) + (K(m|m_1^*) + K(m_1)) - O(1) \\
&= K(k|m_1^*) + K(m|m_1^*) - K(k, m|m_1^*) - O(1) \\
&= I_K(k : m|m_1^*) - O(1).
\end{aligned}$$

□

Chapter 4

Analysis of Commitment Systems

In Chapter 3, we have analysed the individual security of cipher and authentication systems. This chapter is devoted to commitment systems. Instead of analysing individual instances of such systems, we show how these can be simply seen as a composition of cipher and authentication systems. The results in this chapter appeared in [PSMA07]. The main results in this chapter are (Theorem 4.1.4) that an unconditionally secure commitment scheme with trusted initializer can be built from

- a composition of an unconditionally secure authentication code without secrecy, without splitting and with no arbitration
- and an unconditionally secure cipher system.

and this decomposition is unique for optimal commitment systems (Theorem 4.3.7). This relation suggests an attack already referred in [NMQO$^+$03] that is the counterpart of the impersonation attack of an authentication system. We give two lower bounds for this attack, one based on entropy and the other merely on counting, similar to Stinson's bounds for authentication. The first of these bounds is already present in [NMQO$^+$03] but while their proof used techniques from hypothesis testing, ours uses only the definition of mutual information and the log sum inequality.

We begin by analysing the possible attacks against commitment schemes in Section 4.1 and then show how these can be built from a cipher system and an authentication code. In Section 4.3, we define the notion of optimal commitment scheme and show that such a scheme is a resolvable design commitment scheme as proposed in [BMSW02]. We follow with the main results of this chapter, showing that optimal systems must be resolvable design commitment schemes and that all of these can be decomposed into

a cipher system and an authentication code. We then propose a generalization of the affine plane commitment scheme in [BMSW02] that is efficiently implementable in both hardware and software by allowing an alphabet of source states with size $|S| = 2^n$ rather than having $|S| = p$ for prime p. In the former case, the needed arithmetic operations reduce to bit shifts and bitwise logical operations, which have very fast hardware and machine-code implementations. We show that this is possible for every n, by building an appropriate Transversal Design and using a result due to Stinson ([Sti87]), to turn it into an unconditionally secure authentication system. Our commitment scheme follows from composition with the one-time pad cipher system. By the previous results, this scheme is optimal.

4.1 Analysis of Commitment Schemes

This section presents an analysis of the possible attacks against a commitment scheme and shows how to build such schemes from a cipher and an authentication scheme.

4.1.1 Security

In a commitment scheme, both participants can launch attacks.

Bob's Attack The attack available to Bob is uncovering Alice's commitment before she reveals her secret. The security of a commitment scheme can be measured by the probability that Alice has of cheating Bob, while at the same time imposing that Bob can not guess Alice's commitment with more than a priori probability. This means Bob's chances at guessing each x should not be altered by his knowledge of v and y, i.e., for all triples (x, y, v), $p(x|y, v) = p(x)$, which can be summarized using Shannon's entropy with $H(X) = H(X|Y, V)$.

Alice's Attacks Suppose Alice commits to a value x and sends $y = f(k, x)$ to Bob, where k is her secret key. Alice cheats Bob if she can reveal a $k' \neq k$ such that $f_{k'}^{-1}(y) = x'$ with $x' \neq x$, and Bob accepts k' as valid, i.e., $g_v(k') = 1$. It is proved in [BMSW02] that a commitment scheme can not be invulnerable against all of Alice's attacks: Alice can compute the set $\mathcal{V}_k = \{v \in \mathcal{V} : p(v|k) > 0\}$ of all the tags that Bob may have. She then picks the tag $v_0 \in \mathcal{V}_k$ that maximizes $p(v|k)$. Let $\alpha = p(v_0|k)$. By an averaging argument, $\alpha \geq 1/|\mathcal{V}_k|$. Now, Alice picks two values $x \neq x'$ and computes

$y = f(k, x)$. But by the concealing property, there is a key k' such that $f(k', x') = y$ and $g(v_0, k') = 1$ which allows Alice to cheat successfully if Bob's tag is v_0. The success probability of this attack is the probability that Bob is holding the tag chosen by Alice, α. It is shown in the same paper that the average probability of this attack is at least $2^{-H(V|K)}$ and therefore there's at least one instance with at least this probability of success.

The attack described above is the counterpart to a substitution attack in an authentication system. There is yet another attack that Alice can perform, which has been pointed in [NMQO+03]. This is the counterpart of an impersonation attack. These relations are a consequence of the construction of commitment schemes from authentication codes. In the previous attack, Alice makes the best possible use of her private information, but she can also launch an attack ignoring it altogether. To do this, Alice simply computes for each key the probability that Bob accepts it, i.e., for a fixed key k she finds

$$\gamma(k) = \sum_{v \in \mathcal{V}_k} p(v). \tag{4.1}$$

She then picks the key that maximizes the above sum and reveals it to Bob in the revealing step. We give two combinatorial lower bounds for this attack when the distribution of the keys and tags is uniform.

Theorem 4.1.1 *There is some $k \in \mathcal{K}$ with probability of success $\gamma(k) \geq \frac{E(|K_v|)}{|\mathcal{K}|}$, where $E(|K_v|)$ signifies the average number of keys that each tag validates.*

Proof: Consider $\gamma(k)$ as defined above. Its average value is

$$1/|\mathcal{K}| \sum_{k \in \mathcal{K}} \sum_{v \in \mathcal{V}_k} p(v) =$$
$$1/|\mathcal{K}| \sum_{v \in \mathcal{V}} |\mathcal{K}_v| p(v) =$$
$$\frac{E(|K_v|)}{|\mathcal{K}|}.$$

Then, by an averaging argument, there is some k which has $\gamma(k) \geq \frac{E(|K_v|)}{|\mathcal{K}|}$. □

Corollary 4.1.2 *There is some $k \in \mathcal{K}$ with probability of success $\gamma(k) \geq \frac{E(|\mathcal{V}_k|)}{|\mathcal{V}|}$, where $E(|\mathcal{V}_k|)$ signifies the average number of tags that validate each key.*

Proof: It suffices to note that $\sum_{k \in \mathcal{K}} |\mathcal{V}_k| = \sum_{v \in \mathcal{V}} |\mathcal{K}_v|$. □

We can show an analog result with Shannon's entropy:

Theorem 4.1.3 *There is some $k \in \mathcal{K}$ with probability of success*

$$\gamma(k) \geq 2^{-I(K;V)}.$$

Proof: By definition of mutual information (see [CT91])

$$-I(K;V) = \sum_{k\in\mathcal{K}, v\in\mathcal{V}} p(k,v) \log \frac{p(k)p(v)}{p(k,v)}.$$

For each $k \in \mathcal{K}$, $p(k,v) = 0$ for every $v \notin \mathcal{V}_k$. Thus, the above can be written

$$\sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k,v) \log \frac{p(k)p(v)}{p(k,v)}. \tag{4.2}$$

The log sum inequality (see [CT91]) states that

$$\sum_{i=1}^{n} a_i \log \frac{b_i}{a_i} \leq \left(\sum_{i=1}^{n} a_i\right) \log \frac{\sum_{i=1}^{n} b_i}{\sum_{i=1}^{n} a_i}. \tag{4.3}$$

Applying (4.3) to (4.2),

$$\begin{aligned}
-I(V;K) &= \sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k,v) \log \frac{p(k)p(v)}{p(k,v)} \\
&\leq \left(\sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k,v)\right) \log \frac{\sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k)p(v)}{\sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k,v)} \\
&= 1 \cdot \log \sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{V}_k} p(k)p(v) \\
&= \log \sum_{k\in\mathcal{K}} p(k)\gamma(k) \\
&= \log E(\gamma(K))
\end{aligned}$$

where $E(\gamma(K))$ is the average value of the success probability for each k. By an averaging argument, there is at least one k that has probability greater or equal to the average value. For this k: $\gamma(k) \geq E(\gamma(K)) \geq 2^{-I(K;V)}$. □

A commitment scheme is said to be unconditionally secure if it is perfectly concealing and the maximum probabilities of success for these two attacks are equal and meet the lower bounds. This implies $H(K) = H(V|K) + H(K|V)$.

4.1.2 Construction of Commitment Schemes

This section presents a proof that an unconditionally secure commitment scheme can be built using an unconditionally secure cipher system and an unconditionally secure authentication system without secrecy as building blocks. Each user in these systems has a function to play. We call that function a "role". In composing a commitment scheme with two different systems, the users of the former will have to play the different roles of the latter at different steps, so we refer to these roles by writing the abbreviation of the system followed by the role played, all within square brackets. In the remainder of the paper, CP stands for "cipher system", AC stands for "authentication system" and CM stands for "commitment system". The cipher system consists of three roles: [CP.Alice], [CP.Bob] and [CP.Eve]; the authentication system has roles [AC.Alice], [AC.Bob] and [AC.Oscar].

As previously mentioned, in a commitment scheme there are two kinds of attacks. The first is against secrecy: Bob must not learn the secret value Alice committed to before the right time, so Alice sends it enciphered. The second attack is against authentication: Alice must not send a fake opening key, so she must send it through an authentication scheme. In the first step, Alice uses a cipher system without receiver. She merely sends Bob an encrypted message, but he must not be able to open it. Essentially, Bob takes the role of [CP.Eve]. After Bob receives a key in the revealing step, he takes the role of [CP.Bob] and opens the ciphertext learning Alice's commitment. In the second step, Alice sends Bob the key to open the encrypted message he has, but Bob needs to be sure it is the right key, distributed to her in the initial phase. Essentially, Alice acts as a relay between Ted and Bob. Since she reads what the initializer sends and has a choice of relaying that message or changing it for another one altogether, she has complete control over the channel. In this phase, Ted plays the role [AC.Alice], Alice plays the role [AC.Oscar] and Bob plays the role [AC.Bob]. We summarize the above in Table 4.1.

User	**Committing Step**	**Revealing Step**
Alice	[CP.Alice]	[AC.Oscar]
Bob	[CP.Eve]	[CP.Bob] / [AC.Bob]
Ted	...	[AC.Alice]

Table 4.1: Roles Played

Theorem 4.1.4 *Given an unconditionally secure cipher system* $CP(\mathcal{P}, \mathcal{C}, \mathcal{K}, f(k,p))$

and an authentication system without secrecy $AC(\mathcal{S}, \mathcal{A}, \mathcal{E}, h(e,s), g(e,\langle s,a\rangle), \alpha, \beta)$ *with* $\mathcal{S} = \mathcal{K}$, *there is a commitment scheme with initializer (per Rivest's model)* $CM(\mathcal{P}, \mathcal{C}, \mathcal{S} \times A, \mathcal{E}, f(s,p), g(e,\langle s,a\rangle), \alpha, \beta)$.

Proof: The several components of the commitment scheme are obtained from the cipher and the authentication system as shown in Table 4.2. Because we have used

Cipher	Commitment	Authentication
$\mathcal{P}$	$\mathcal{X}$	...
$\mathcal{C}$	$\mathcal{Y}$	...
$\mathcal{K}$	...	$\mathcal{S}$
...	$\mathcal{K}$	$\mathcal{S} \times \mathcal{A}$
...	$\mathcal{V}$	$\mathcal{E}$
$f(k,p)$	$f(k,x)$	...
...	$g(k,v)$	$g(k,\langle s,a\rangle)$

Table 4.2: Equivalences between systems

letters given in the initial definitions, there are two different alphabets labelled $\mathcal{K}$. They are not to be confused. For each $k \in CP.\mathcal{K}$, there are $|\mathcal{A}|$ different $k' \in CM.\mathcal{K}$, all with the same behaviour in the cipher system. Considering the analysis in [BMSW02], these $|\mathcal{A}|$ keys form a parallel class of keys in the combinatorial design used as basis for the resolvable design commitment scheme. The protocol is as follows:

1. **Initialization:** Ted chooses uniformly at random an encryption key $s \in \mathcal{S}$ and an authentication key $e \in \mathcal{E}$, computes the authenticator $a = h(e,s)$ and sends $\langle s,a\rangle$ to Alice and e to Bob.

2. **Committing Step:** Alice commits to x by sending Bob the encryption $y = f(k,x)$.

3. **Revealing Step:** Alice sends Bob a possibly false key $\langle s',a'\rangle$. Bob checks if $g_e(\langle s',a'\rangle) = 1$ and if so, he decrypts $x' = f_{s'}^{-1}(y)$.

This construction yields a commitment scheme that follows Rivest's model, as is shown next. The crucial point of this construction is that the source value that Ted wants to send Bob in the revelation step is the actual key that the latter must use to open Alice's commitment. The figures in Section 4.2 can help to understand this. It is easy to verify that the families of functions $f(k,p)$ and $g(k,\langle s,a\rangle)$ satisfy the formal

requirements of the commitment scheme. Now we check the concealing and binding properties.

Concealing Let x_0 be the value Alice committed to, k_0 the key she holds and $y_0 = f(k_0, x_0)$ the value Bob received. Let e_0 be Bob's tag. Let $X_{y_0} = \{x \in \mathcal{X} : \exists k \in \mathcal{K} \text{ s.t. } f(k,x) = y_0\}$ be the set of possible plaintexts for the ciphertext Bob holds, $\mathcal{K}_{e_0} = \{k \in \mathcal{K} : \exists a \in \mathcal{A} \text{ s.t. } h(e_0, k) = a\}$ be the set of possible Alice's keys and let $X_{y_0,e_0} = \{x \in \mathcal{X} : \exists k \in \mathcal{K}_{e_0} \text{ s.t. } f(k,x) = y_0\}$ be the set of possible values for Alice's commitment given the information Bob knows. Because the authentication system does not have splitting, for each possible $k \in \mathcal{K}$ and $e \in \mathcal{E}$, there is exactly one $a \in \mathcal{A}$ such that $h(e,k) = a$. Therefore, all the keys can be associated to each particular e, and so $\mathcal{K} = \mathcal{K}_{e_0}$, implying that $X_{y_0} = X_{y_0,e_0}$.

Bob's probability of guessing Alice's commitment without or with knowledge of e_0 is, respectively, $p(x_0|y_0) = \sum_{k \in \mathcal{K}, f(k,x_0)=y_0} p(k)$ and $p(x_0|y_0,e_0) = \sum_{k \in \mathcal{K}_{e_0}, f(k,x_0)=y_0} p(k)$ and by the above reasoning they're equal. Then

$$\begin{aligned} H(X|Y,E) &= E_{y,e}(H(X|Y=y, E=e)) \\ &= E_{y,e}(\sum_{x \in \mathcal{X}} p(x|y,e) \log 1/p(x|y,e)) \\ &= E_y(\sum_{x \in \mathcal{X}} p(x|y) \log 1/p(x|y)) \\ &= E_y(H(X|Y=y)) \\ &= H(X|Y). \end{aligned}$$

If follows, by assumption, that $H(X|Y,E) = H(X)$ and this commitment scheme satisfies the concealing property.

Binding Let x_0 be the value Alice committed to and (k_0, a_0) be the key/authenticator pair that she holds. In order to reveal a value $x' \neq x_0$, Alice needs to make Bob accept a key $k' \neq k_0$. Alice can make two kinds of attack, as described in Section 4.1.1. Her chances of success are, for the first attack:

$$\max_{(k',a') \in \mathcal{K} \times \mathcal{A}} \sum_{e \in \mathcal{E},\, g(e,(k',a'))=1} p(e).$$

This corresponds to the impersonation attack against the authentication system and so this probability is at most α. Likewise, for the second attack:

$$\max_{(k',a'),(k_0,a_0) \in \mathcal{K} \times \mathcal{A}, k' \neq k_0} \frac{\sum_{e \in \mathcal{E},\, g(e,(k',a'))=1,\, g(e,(k_0,a_0))=1} p(e)}{\sum_{e \in \mathcal{E},\, g(e,(k',a'))=1} p(e)},$$

which corresponds to a substitution attack and is at most β. Thus, this commitment is $(1 - \max(\alpha, \beta))$-binding. □

Corollary 4.1.5 *Given an unconditionally secure cipher system and an unconditionally secure authentication system without secrecy there is an unconditionally secure commitment scheme with initializer (per Rivest's model).*

Next, we show how the different flows of information in the three systems are related.

4.2 Flow Analysis

In the conclusion to the paper [BMSW02], the authors suggest a possible relation between commitment schemes and authentication schemes with arbitration, but point that the information flows between these systems are different. Here, we analyse the different flows of information in a commitment scheme, and how these are realized through the flows present in the cipher and in the authentication systems. We understand by information flows the data that is sent from one user to another user. The following pictures help visualize the flows in the different systems. In these figures, there are blocks representing each participant in the system and arrows representing the messages sent by them, this is, the flows of information between users. Within some blocks is another name within square brackets. This represents the name of the user of the commitment scheme that will be playing the role indicated by the block. For instance, when Alice sends her commitment to Bob, he is playing the role of Eve in the cipher scheme: he receives a ciphertext but can not read it. Next is shown

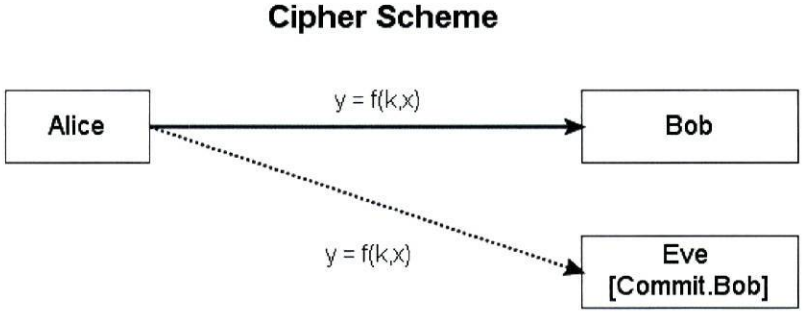


Figure 4.1: A Cipher Scheme

how each flow is used to implement the flows of the final commitment scheme. When

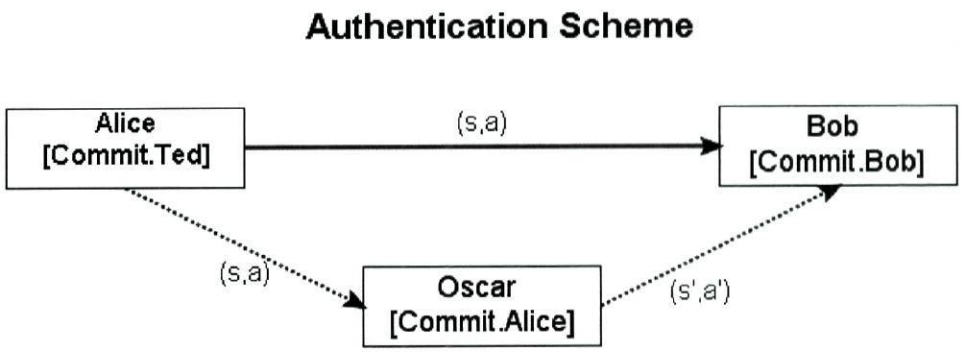


Figure 4.2: An Authentication Scheme

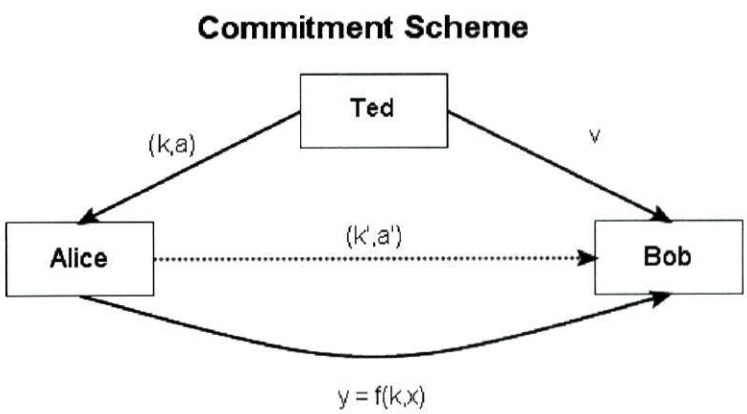


Figure 4.3: A Commitment Scheme

necessary, we describe roles with the same notation of Section 4.1.2. In the following list, we describe the flows in each system and identify them with a name between brackets that indicates the system the flow belongs to and the step when it takes place. When some steps have two similar flows, these are further distinguished with letters 'a' and 'b'.

(CP1) Alice (a) and Bob (b) receive a secret key by some secure channel. This includes the case where they create a key themselves and exchange it.

(CP2) A message is sent from Alice to Bob (a) and possibly also read by Eve (b).

In an authentication system as described above, there are the following flows:

(AC1) Alice (a) and Bob (b) receive a secret key by some secure channel.

(AC2) A message is sent from Alice to Oscar (a), who may change it before relaying it to Bob (b). Note this is just a simplified model. In reality, Alice sends the message to Bob, but Oscar may intercept and alter it or not.

In a commitment scheme, there are the following information flows:

(CM1) Ted gives a key to Alice (a) and a verification tag to Bob (b).

(CM2) Alice sends her commitment to Bob.

(CM3) Alice sends her key to Bob to open her commitment.

The information flows of the commitment scheme are carried out by the information flows of the other systems like this:

- Flow (CM1.b) is achieved by flow (AC1.b). Flow (AC1.a) is ignored because Ted does not need to remember the key after he creates a valid message to send Alice. Flow (CM1.a) is achieved by flow (AC2.a), that is, Ted takes the role [AC.Alice] and sends a message to Alice ([AC.Oscar]). Due to the nature of the construction, namely because the authentication system does not have secrecy, flow (AC2.a) includes flow (CP1.a), because Alice now has a key for the cipher system.
- Flow (CM2) is achieved by flow (CP2.b).
- Flow (CM3) is achieved by flow (AC2.b). From this message, Bob deduces a key, completing flow (CP1.b), and opens the commitment by flow (CP2.a).

Cipher	Commitment	Authentication
(CP1.a)	(CM1.a)	(AC2.a)
...	(CM1.b)	(AC1.b) [(AC1.a)]
(CP2.b)	(CM2)	...
(CP1.b) (CP2.a)	(CM3)	(AC2.b)

Table 4.3: Information Flows

4.3 Optimal Commitment Schemes

In [BMSW02], the authors propose a general commitment scheme which they call "resolvable design commitment scheme". In this section, we define optimal commitment schemes and show that they are resolvable design affine commitment schemes. Then, we close the circle showing that all resolvable design commitment schemes can be viewed as the composition of a cipher system and an authentication system. For simplification, in what follows, consider that the source values, keys and verification tags are distributed uniformly, since this maximizes uncertainty and therefore security.

Definition 4.3.1 *A commitment scheme* $CM(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{V}, f(k,x), g(v,k), \alpha, \beta)$ *is optimal if it is unconditionally secure,* $|\mathcal{X}| = |\mathcal{Y}|$ *and has the minimum number of keys for a fixed number of source states and the desired security level. Besides, the probability of Alice's cheating should be equal to the probability of Bob's cheating.*

Lemmas 4.3.2 and 4.3.3 give some properties that an optimal commitment system must have. Lemma 4.3.4 excludes BIBDs as the possible minimal system, and this is necessary because such systems are not resolvable. This means that there can be pairs of blocks with empty intersection and so these are counted in Lemma 4.3.5. After these lemmas, we're ready to give the two main theorems: that an optimal commitment system must be affine resolvable and that a resolvable commitment scheme can be decomposed into a cipher system and an authentication system.

Lemma 4.3.2 *If a commitment scheme* $CM(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{V}, f(k,x), g(v,k), \alpha, \beta)$ *is optimal, then* $\alpha = \beta$ *and* $|\mathcal{V}| = (1/\alpha)^2$.

Proof: By definition,

$$\alpha = \max_{k \in \mathcal{K}} |\mathcal{V}_k|/|\mathcal{V}| \geq E(|\mathcal{V}_k|)/|\mathcal{V}|,$$

and by optimality of the system the above holds with equality. Then, $|\mathcal{V}_k|$ is constant for all k. Likewise, and by Theorem 4.1.1, $|\mathcal{K}_v|$ must be constant. By definition,

$$\begin{aligned}\beta &= \max_{k \neq k' \in \mathcal{K}} \frac{\Pr[g(v,k)=1, g(v,k')=1]}{\Pr[g(v,k')=1]} \\ &= \max_{k \neq k' \in \mathcal{K}} \frac{|\{v \in \mathcal{V} : g(v,k) = g(v,k') = 1\}|}{|\{v \in \mathcal{V} : g(v,k) = 1\}|} \\ &= \max_{k \neq k' \in \mathcal{K}} \frac{|\mathcal{V}_k \cap \mathcal{V}_{k'}|}{|\mathcal{V}_k|}.\end{aligned}$$

Let $\mu = \max_{k \neq k' \in \mathcal{K}} |\mathcal{V}_k \cap \mathcal{V}_{k'}|$. Then

$$\beta = \frac{\mu}{|\mathcal{V}_k|}.$$

The value μ can not be 0 because if so each tag would verify exactly one key and Bob would be able to cheat Alice with absolute certainty, so β is minimum if $\mu = 1$ and

$$\begin{aligned}\alpha &= |\mathcal{V}_k|/|\mathcal{V}|, \\ \beta &= 1/|\mathcal{V}_k|.\end{aligned}$$

We show that γ is minimum when $\alpha = \beta$. Let $|\mathcal{V}_k|$ and $|\mathcal{V}|$ be such that $\alpha = \beta = \gamma$. Then $|\mathcal{V}_k|^2 = |\mathcal{V}|$. Denote by α_0 and γ_0 the values of α and γ respectively in this case. Assume for contradiction that there is some combination of values $|\mathcal{V}_k|$ and $|\mathcal{V}|$ such that $\gamma = \max(\alpha, \beta) < \gamma_0$. Assume w.l.o.g that $\alpha > \beta$. Then

$$\begin{aligned}&\alpha = \gamma < \gamma_0 = \alpha_0 \Rightarrow \\ &|\mathcal{V}_k|/|\mathcal{V}| < |\mathcal{V}|^{1/2}/|\mathcal{V}| \Leftrightarrow 1/|\mathcal{V}_k| > 1/|\mathcal{V}|^{1/2} \Rightarrow \\ &\beta > \alpha_0 > \alpha.\end{aligned}$$

Thus we have a contradiction, and so the minimum γ is achieved when $\alpha = \beta$. This implies that $1/|\mathcal{V}_k| = |\mathcal{V}_k|/|\mathcal{V}|$ and $|\mathcal{V}| = |\mathcal{V}_k|^2 = (1/\alpha)^2$. □

Lemma 4.3.3 *If a commitment scheme* $CM(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{V}, f(k,x), g(v,k), \alpha, \beta)$ *is optimal, then* $|\mathcal{X}|^2 = |\mathcal{K}| = |\mathcal{V}|$.

Proof: Let v be Bob's tag and y the coding of Alice's commitment. Define $\mathcal{X}_{y,v} = \{x : \exists\, k \in \mathcal{K}_v \; s.t. \; f_k^{-1}(y)\}$ to be the set of possible commitments for Alice given the information Bob holds. Then, $|X_{y,v}| \leq |\mathcal{K}_v|$. Because $|\mathcal{K}_v|$ is constant,

by Lemma 4.3.2,

$$\begin{aligned} H(X|Y,V) &= \sum_{y\in\mathcal{Y},v\in\mathcal{V}} \Pr(y,v)H(X|Y=y,V=v) \\ &\leq \sum_{y\in\mathcal{Y},v\in\mathcal{V}} \Pr(y,v)\log|X_{y,v}| \\ &\leq \sum_{y\in\mathcal{Y},v\in\mathcal{V}} \Pr(y,v)\log|\mathcal{K}_v| \\ &= \log|\mathcal{K}_v| \end{aligned}$$

and by optimality of the system $\log|\mathcal{K}_v| \geq H(X) = \log|\mathcal{X}|$.

By the proof of Corollary 4.1.2, $|\mathcal{K}| \cdot |\mathcal{V}_k| = |\mathcal{K}_v| \cdot |\mathcal{V}|$ and using Lemma 4.3.2, this brings $|\mathcal{K}| = |\mathcal{K}_v| \cdot |\mathcal{V}_k| \geq |\mathcal{X}| \cdot |\mathcal{V}_k|$. But since the system is optimal and the number of keys is minimal, then it must be that $|\mathcal{K}_v| = |\mathcal{X}|$. Bob's chance of guessing Alice's commitment is $1/|\mathcal{X}|$, by definition of security. From Lemma 4.3.2, Alice's chance is $1/|\mathcal{V}_k|$. Because in an optimal system Alice's chance of success is equal to Bob's, $1/|\mathcal{X}| = 1/|\mathcal{V}_k| \Leftrightarrow |\mathcal{X}| = |\mathcal{V}_k|$ which implies $|\mathcal{K}| = |\mathcal{X}|^2 = |\mathcal{V}|$. □

Lemma 4.3.4 *If a commitment scheme $CM(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{V},f(k,x),g(v,k),\alpha,\beta)$ is optimal, then its incidence matrix can not be a Balanced Incomplete Block Design (BIBD).*

Proof: By Lemma 4.3.3, $|\mathcal{V}| = |\mathcal{K}|$. Suppose the incidence matrix of the commitment scheme is a $2-(v,k,\lambda,b,r)$ Balanced Incomplete Block Design. Then the design is symmetric and by Theorem 2.14 in [AH97], any two keys have exactly λ tags validating them. By the proof of Lemma 4.3.2, the maximum intersection between any two lines should be 1, so $\lambda = 1$. Then, by Theorem 2.6.2, each tag validates exactly $r = (v-1)/(k-1)$ keys and by Theorem 2.14 in [AH97], $r = k$. This brings $b = v = k^2 - k + 1$ where $b = |\mathcal{K}|$ and $v = |\mathcal{V}|$. By definition, $r = |\mathcal{K}_v|$.

By Lemma 4.3.3, $|\mathcal{K}|/|\mathcal{X}| = |\mathcal{K}_v|$. Then, b/k must be an integer, but

$$\begin{aligned} \frac{b}{k} &= \frac{k^2-k+1}{k} \\ &= k - 1 + 1/k \end{aligned}$$

and this can not be an integer for $k > 1$. □

Lemma 4.3.5 *Let $p = |\mathcal{X}|$. If a commitment scheme $CM(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{V},f(k,x),g(v,k),\alpha,\beta)$ is optimal, then the sum of distinct pairs of keys that don't have any tag in common is $p^2 \cdot (p-1)/2$.*

Proof: Consider a square matrix where cell (i, j) contains the value $|\mathcal{V}_{k_i} \cap \mathcal{V}_{k_j}|$ where $k_i \neq k_j \in \mathcal{K}$. There are $p^2 \cdot (p^2 - 1)$ filled cells. We count the pairs that have a non-empty intersection. Each key has tag v_1 in common with $p - 1$ different keys and since it is validated by p tags, each key contributes with $p(p - 1)$ to the total of the sum

$$\sigma' = \sum_{k_i \in \mathcal{K}} \sum_{k_j \neq k_i \in \mathcal{K}} |\mathcal{V}_{k_i} \cap \mathcal{V}_{k_j}|.$$

Therefore, $\sigma' = p^3(p - 1)$. But this sum counts each pair twice, so the total number of distinct intersections is $\sigma = p^3(p - 1)/2$. We can find the total number of distinct key pairs that don't intersect, recalling that in this particular case all filled cells are either 0 or 1, which means that σ is the sum of all 1s in the table. As before, only a half of the matrix needs to be considered. Then, the number of distinct key pairs without tags in common is

$$(p^4 - p^2)/2 - p^3(p - 1)/2 = p^2 \cdot (p - 1)/2.$$

□

Theorem 4.3.6 *If a commitment scheme $CM(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{V}, f(k, x), g(v, k), \alpha, \beta)$ is optimal, then it is an affine resolvable commitment scheme, and all keys in each parallel class encrypt each value $x \in \mathcal{X}$ to the same value $y \in \mathcal{Y}$. That is, the function $y = f(k, x)$ depends only on the index of the parallel class containing k, and not on k itself.*

Proof: Let $p = |\mathcal{X}|$. From the previous results, there are p^2 keys. Fix some $x_0 \in \mathcal{X}$. The concealing property implies that there must be exactly p keys transforming x_0 into each possible value $y \in \mathcal{Y}$. Then, the keys can be grouped in p groups such that all keys $k_{i,j}$ in group i satisfy $f(k_{i,j}, x_0) = y_i$. There are exactly p keys validated by each verifier tag. For each two keys k_i and k_j validated by the same tag, it must happen that $f(k_i, x_0) \neq f(k_j, x_0)$ or else there won't be enough keys to hit all the values in $\mathcal{Y}$. But by Lemma 4.3.5 and a counting argument, this implies that all pairs of keys in different groups must have exactly one common tag. Since there are p disjoint keys in each group, each validated by p tags, each group forms a partition of $|\mathcal{V}|$ and is therefore a parallel class. The design is therefore resolvable and since the maximum intersection between two keys is $1 = k^2/v$ it is also affine.

Now consider a value $x_1 \in \mathcal{X}$ different from x_0. Suppose there are two keys k_i, k_j in different groups that code x_1 in the same way. That is:

$$f(k_i, x_0) \neq f(k_j, x_0)$$
$$f(k_i, x_1) = f(k_j, x_1)$$
$$|\mathcal{V}_{k_i} \cap \mathcal{V}_{k_j}| = 1.$$

Following the same reasoning as above, if $f(k_i, x_1) = f(k_j, x_1)$ then they can not have any tag in common, contradicting the previous division in groups. Therefore, keys in different groups code x in different ways and by a counting argument all keys in the same group transform x into the same y. Repeating the argument for any $x_l \in \mathcal{X}$ and for all groups, it must happen that all keys k_i, k_j in the same group satisfy

$$f(k_i, x_l) = f(k_j, x_l)$$

and so the theorem is proved. □

Theorem 4.3.7 *A resolvable design commitment scheme obtained from a resolvable* $1-(v,k,\lambda,b,r)$ *design* $(\mathcal{D},\mathcal{S})$ *is a composition of a perfectly secure cipher system and an authentication code with* $P_{d0} = k/v$ *and* $P_{d1} = \frac{\max|B_1 \cap B_2|}{k}$ *for all distinct* $B_1, B_2 \in \mathcal{D}$.

Proof: Let B be Alice's block and w be Bob's tag in the above system. Then $w \in B$. By Theorem 2.6.2, $b = r \cdot v/k$. Then, w can be seen as a number between 0 and $v-1$.

By definition of resolvable design, B belongs to some parallel class. Then, B can be written (i,j), where i is the index of the parallel class and j is the index of the block within the parallel class. The pair (i,j) can be interpreted as a pair source value / authenticator.

When Alice commits to $x_0 \in \mathbb{Z}_r$, she sends Bob $y_0 = (x_0 + i) \bmod r$. Then, x_0 can be seen as a symbol in alphabet $\Sigma = \mathbb{Z}_r$ and y_0 as a displacement of i positions over that alphabet. This corresponds to applying a Ceasar's cipher to the secret message x_0. In general, Ceasar's cipher is not secure, but here the message is composed of only one symbol, which means its size is equal to the size of the key. In this situation, it is equivalent to the one-time pad and is unconditionally secure. Since all blocks in the same parallel class encipher x_0 in exactly the same way, the index of the parallel class represents the cipher key used by Alice.

Now we show that the design used to check the validity of the value revealed after the commitment can also be used to make an authentication scheme with r source states,

v/k possible authenticators for each source state and v encoding rules. Any block from the design can be identified by a pair (i, j), where i is the index of the parallel class it belongs to and j its index within that class. Let B be the set represented by a block with indices (i, j). Then, B can be interpreted as belonging to an authentication code: i represents the source code and j the respective authenticator. Each element $w \in B$ contributes for the definition of an encoding rule in the following way: $f(w, i) = j$. Since each w belongs to exactly one block in each parallel class, there is a unique value associated to each pair (w, i). Since there are k elements in each block, there are exactly k encoding rules associating i to j.

The probability of an impersonation attack is the maximum probability of finding the right authenticator for a specific pair key / source state. There are a total of v encoding rules. The number of rules that associate some source state i to a given authenticator j is, by construction, k. Thus, for all such pairs (i, j),

$$\text{payoff}(i, j) = \sum_{w \in \mathcal{S}: f(w,i)=j} 1/v = k/v$$

and because this is constant for all pairs, it is the probability of an impersonation attack. For the substitution attack, suppose the attacker knows that the secret encoding rule associates i_1 to j_1. By construction, there are k such keys. For any pair (i, j), the probability of success is

$$\begin{aligned} \text{payoff}(i, j, i_1, j_1) &= \frac{\sum_{w \in \mathcal{S}: f(w,i)=j, f(w,i_1)=j_1} p(w)}{\sum_{w \in \mathcal{S}: f(w,i_1)=j_1} p(w)} \\ &= \frac{|B_{i,j} \cap B_{i_1,j_1}|}{k} \end{aligned}$$

where $B_{i,j}$ is the block of the design indexed by (i, j). Then, the probability of the substitution attack is $\frac{\max |B_1 \cap B_2|}{k}$ over all $B_1, B_2 \in \mathcal{D}, B_1 \neq B_2$. □

4.4 Generalization to Galois Fields

As noted in [BMSW02], affine resolvable 2-designs are optimal among the resolvable designs in terms of binding probabilities, however not many classes of such designs are known to exist. Notwithstanding this, there are other kinds of designs that can achieve the same goals, namely Transversal Designs.

This section addresses this question by showing how to construct a resolvable Transversal Design $TD(2^n, 1, 2^n)$, for any n, that is also a $1-(2^{2n}, 2^n, 1)$ affine resolvable design.

From such a design, we then build an unconditionally secure authentication code and an unconditionally secure commitment scheme.

Theorem 4.4.1 *For any positive integer n, it is possible to construct a Transversal Design $TD(2^n, 1, 2^n)$.*

Proof: The order of any finite field can be written p^k, where p is a prime and $k \geq 1$ is an integer (see [LN83]). Let $GF(p^n)$ represent one such field. For any finite field $GF(p^n)$, there is a primitive polynomial of degree n and coefficients modulo p ([LN83]). For composite orders p^n, the elements of the field are considered to be polynomials and the operations of the field are addition and multiplication of polynomials modulo the prime p. Addition is denoted by $\oplus$ and multiplication by $\odot$. Fix some n and some primitive polynomial for $GF(2^n)$. We build a table with 2^{2n} rows and 2^n columns. The rows are divided in 2^n groups of 2^n elements each, and uniquely identified by a pair (a, b) where $a, b \in \mathbb{Z}_{2^n}$. Each column is labelled by a number $x \in Z_{2^n}$. Cell $((a, b), x)$ holds the value $a \odot x \oplus b$, which is a number in $\mathbb{Z}_{2^n}$.

We show this table represents a $TD(2^n, 1, 2^n)$ transversal design. Consider the set of points $\mathcal{V} = \{0, 1, \ldots, 2^{2n} - 1\}$ and divide them in 2^n groups of 2^n points. Each row represents a block with 2^n points, one from each group. For row j, the i^{th} value represents the index of the point in the i^{th} group that belongs to the j^{th} block. By construction, each block has one point in each group. Finally, each two points from distinct groups can occur in only one block. To see this, let (x_0, y_0) and (x_1, y_1) be two points from distinct groups, where $x_0 \neq x_1$. For both points to be in the same block, there must be a pair (a, b) such that $a \odot x_0 \oplus b = y_0$ and $a \odot x_1 \oplus b = y_1$. Then,

$$a \odot x_0 - y_0 = a \odot x_1 - y_1 \Leftrightarrow$$
$$a \odot (x_0 - x_1) = y_0 - y_1.$$

Since $(x_0 - x_1)$ and $(y_0 - y_1)$ are defined and $(x_0 - x_1) \neq 0$, then a is completely determined, and so is b. That means there is only one pair satisfying both equations, which means both points can belong to only one block. This concludes the proof. □

This design originates an authentication code $AC(2^n, 2^n, 2^{2n}, 1/2^n, 1/2^n)$, as proved in [Sti87]. With such an unconditionally secure authentication scheme, and using the one-time pad as a perfect cipher system, we can build an unconditionally secure commitment scheme as outlined in section 4.1.2. Let $\mathcal{S} = \{0, 1\}^n$. The protocol is as follows:

1. **Initialization:** Ted chooses randomly a pair $(a, b) \in \mathcal{S} \times \mathcal{S}$ and a number $x_1 \in \mathcal{S}$, computes $y_1 = a \odot x_1 \oplus b$ and sends the pair (x_1, y_1) to Bob and the pair (a, b) to Alice.

2. **Committing Step:** Alice commits to $x_0 \in \mathcal{S}$ by enciphering $y_0 = x_0 \oplus a$ and sending y_0 to Bob.

3. **Revealing Step:** Alice sends (a', b') to Bob, who checks that $a' \odot x_1 \oplus b' = y_1$. If so, he computes $x_0' = y_0 \oplus a'$ and accepts x_0' as Alice's commitment.

The resulting commitment scheme is a generalization of the affine plane commitment scheme. It uses the one-time pad as cipher system, which is very fast to implement both in software and hardware. Besides, it allows the use of a complete alphabet of strings of size n, whereas in the affine plane with order p, the alphabet of allowed values does not coincide with any alphabet of all strings of a given size. In general, the latter systems will be less efficiently implemented in hardware and software because the basic instructions are more oriented to a fixed size of bits than the corresponding arithmetic value.

Chapter 5

Computational Entropy

In this chapter, we investigate a parallel of the notion of entropy for the computational setting and its relation with a time-bounded version of Kolmogorov complexity. The aim of this approach is to devise an analysis analogue to the one made in Chapter 3 that can be applied to systems which have only proofs of security based on computational assumptions. We could not reach this ideal goal yet, because we were not able to lower bound the expected Kolmogorov complexity of the plaintext given the ciphertext by an appropriate function of computational entropy. But we established the opposite relation and we are confident that further work can close this gap. The results in this chapter appeared in [Pin07] and were accepted for publication in a special issue of the Journal of Theory of Computing Systems.

In the case of information theoretic security, the justification for the use of Kolmogorov complexity was the asymptotic equality between Shannon's entropy and the average value of Kolmogorov complexity, and on the other side the formal similitude of the theory of both quantities. But to analyse cryptographic systems with computational security, we must be able to consider the limitations in the computational power of the attacker. Accordingly, we study a parallel to the information theoretic case in the present chapter, by considering notions of computational entropy and using polynomial time-bounded Kolmogorov complexity and the relations between them.

5.1 Introduction

Randomness is an essential concept in computer science. It is fundamental in cryptography and has been used extensively to provide fast algorithms for otherwise seemingly

difficult problems. Unfortunately, it is hard, if not impossible, for classical computers to produce true random bits. For this reason, it is necessary to simulate randomness by deterministic methods, but the distributions thus generated are only useful if they are 'random enough', or pseudorandom. The definition of pseudorandomness is based on computational indistinguishability from a truly random distribution, as was referred in Chapter 2.

The randomness of a distribution is measured by its entropy, introduced by Shannon in [Sha48]. A related concept is that of minimum entropy (also min-entropy), which is a lower bound on the entropy of some distribution. It is natural to try to extend these objective measures to the computational case, thus defining notions of computational entropy. The first such definition is due to Yao ([Yao82]), but the most used is due to Håstad, Impagliazzo, Levin and Luby ([HILL99]). Barak, Shaltiel and Wigderson provide definitions for computational analogues of min-entropy in [BSW03]. In Information Theory, randomness is tightly related to compression, in the sense that the output of a random distribution cannot be noticeably compressed. Shannon's entropy establishes a bound on how much a random source can be compressed, while at the same time Kolmogorov complexity gives the ultimate bound to the compression that can be achieved for an individual string.

Yao's definition of effective entropy has not been much studied. Goldberg and Sipser ([GS85]) analyse languages that can be compressed efficiently by a probabilistic machine, which is the computation model considered by Yao, but do not mention any notion of computational entropy. The notion of pseudoentropy appears in [HILL99]. In [BSW03], the authors introduce an analogue of min-entropy that closely resembles that in [HILL99]. For this reason, they call it HILL-type pseudoentropy (written $H_\varepsilon^{\mathrm{HILL}}$). They introduce a similar notion, metric pseudoentropy (written $H_\varepsilon^{\mathrm{Metric}}$), and present a measure based on a compression idea which they called Yao-type pseudoentropy (written $H_\varepsilon^{\mathrm{Yao}}$). They show that for some models of computation, metric entropy is equivalent to HILL-type entropy. Wee ([Wee04]) studies compressibility, improving on a result from [GS85] that separated compressibility from pseudoentropy in an oracle setting and giving a separation between BSW's metric pseudoentropy and Yao-type pseudoentropy. Whereas the source used in [GS85] was not samplable by uniform polynomial circuits, Wee demonstrates the existence of an oracle relative to which there exist samplable sources with low-entropy that nevertheless can not be significantly compressed by polynomial size circuits. In Wee's opinion, this result suggests pseudoentropy is not the right lower bound for the size of the compression on polynomially samplable sources. Another paper, by Trevisan, Vadhan and Zuckerman ([TVZ05]),

also continues the work began in [GS85]. They give a method to compress sources that are uniform on their support and have efficient membership oracles that uses a different technique from the one used in [GS85]. The difference between compressibility and indistinguishability is further studied in [HLR07], where the authors define conditional notions of computational min-entropy and show an example of two distributions X and Z such that X has high Yao-type entropy but low HILL-type entropy when conditioned on Z.

This chapter focuses on these measures, comparing Yao-type pseudoentropy, effective entropy and the expected value of the time-bounded Kolmogorov complexity. Effective entropy is by definition the lower bound for the average length of a codeword of any probabilistic compression scheme for a polynomially samplable source and because time-bounded Kolmogorov complexity is the lower bound for a description of any single word in a fixed time, it seems reasonable that its expected value be related to effective entropy. We show that all three are equivalent for the uniform distribution. We also show that for the distribution $\mathbf{m}^t$, Yao-type pseudoentropy and effective entropy have very different values. This result depends on a standard complexity assumption and it can be applied to $H_\varepsilon^{\text{Metric}}$, thus giving a clear separation between pseudoentropy (by a connection of $H_\varepsilon^{\text{Metric}}$ to $H_\varepsilon^{\text{HILL}}$ given in [BSW03]) and effective entropy. We note that if we remove the randomness in the encoder then the result is unconditional.

Our results are as follows: in Section 5.3, we show that Yao's effective entropy (H_c) is at least as large as Yao-type pseudoentropy ($H_\varepsilon^{\text{Yao}}$) when considering efficiency to be represented either by **FP** or **PPT**. For the class **FP**, we do this in two steps: first, a general theorem states that if $H_\varepsilon^{\text{Yao}}$ is larger than a function of n and a parameter i, then H_c will be larger than another function of n and i; then, we show that this implies that H_c is larger than a function of $H_\varepsilon^{\text{Yao}}$. In Section 5.4, it is shown that the effective entropy is at least as large as the average of the resource-bounded Kolmogorov complexity. We prove this relation first for a deterministic version of H_c and then show how it also holds for the original version of H_c if there are hard functions in **EXP**. Section 5.5 shows that H_c is not necessarily larger than Yao-type pseudoentropy, although it is always larger than a variant thereof, by showing a computable distribution that is a counter-example to that relation. In Section 5.6, we consider two specific distributions, uniform and $\mathbf{m}^t$. We show that for the uniform distribution, the three notions are equivalent, while for $\mathbf{m}^t$ there is a difference between the values of metric and Yao-type entropy of [BSW03] on the one hand, and that of the expected Kolmogorov complexity, and under a reasonable complexity assumption, Yao's effective entropy of [Yao82], on the other.

This comparison of computational entropies is new in the literature, and our results regarding the values of these entropies for the universal distribution offer more evidence that with polynomial time bounds, pseudorandomness and maximal compressibility are not measured by the same quantity.

5.2 Preliminaries

Throughout this chapter, we use the same notations used in the original papers: H_c for Yao's effective entropy, $H_\varepsilon^{\text{Metric}}$ and $H_\varepsilon^{\text{Yao}}$, where ε is a real parameter, for BSW's metric and Yao-type pseudoentropies respectively. We now give the definitions of computational entropy used.

5.2.1 Yao's Effective Entropy (H_c)

Definition 5.2.1 *Consider a fixed finite alphabet Σ and the language $L = \Sigma^+$. A source S is a random variable defined over L with probability distribution p subject to the restriction that $\sum_{x \in L} p(x)|x|$ converges. A source ensemble S is a sequence of sources $S_1, S_2, \ldots$ with probability distributions $p_1, p_2, \ldots$ such that for some positive constants t_1, t_2 with $t_2 > t_1$, $p_n(y) > 0$ implies $n^{t_1} < |y| < n^{t_2}$.*

To simplify the notation in the rest of the chapter, we define a constant $\lambda = n^k$ when n and k are understood from the context.

Definition 5.2.2 *Let $S_n^\lambda = \underbrace{S_n \times \cdots \times S_n}_{\lambda \text{ times}}$ be the random variable over $\lambda = n^k$ independent draws of S_n.*

Definition 5.2.3 *A $(t,k)-$encoding scheme for S is a triple of probabilistic polynomial algorithms $M = (M_A, M_B, M_C)$ such that:*

- *M_A receives as input a parameter n and a "text" x composed of λ words of L. These words are independently identically sampled from S_n. Then, $x = (w_1, w_2, \ldots, w_\lambda)$ has probability of occurring equal to $p_n^\lambda(x) = p_n(w_1) \cdot p_n(w_2) \cdot \ldots \cdot p_n(w_\lambda)$, i.e., x is sampled from S_n^λ.*
- *For all sufficiently large n, for all $x \in S_n^\lambda$, $\Pr[M_B(M_A(n,x)) \neq x] < 1/n^t$, where the probability is over the random coins of M_A and M_B.*

- *Let $b > 0$ be any fixed constant, and let $u = O(n^b)$. For all sufficiently large n and all $x_1, x_2, \ldots, x_u$ where each x_i is independently distributed from S_n^λ with probability $p_n^\lambda(x_i)$, $\Pr[M_C(n, z_1 z_2 \cdots z_u) \neq z_1 z_2 \cdots z_u] < 1/n^t$, where $z_i = M_A(n, x_i)$ and the probability is over the random coins of M_A and M_C.*

Informally, M_A is an encoder algorithm for sequences of symbols of S, or put another way symbols of source S_n^λ, into finite binary strings. M_B is the respective decoder. Since both algorithms are probabilistic, a negligible chance of error is allowed in retrieving x from a corresponding encoding.

The last property basically states that to code a whole text we can encode different blocks of it and concatenate the results, because M_C is able to separate its input into the encodings corresponding to the original blocks so that each of them can be correctly decoded. Roughly, the code is uniquely decipherable.

Yao considered that also M_C could fail in breaking up the encodings of different texts, and so he required that this probability were equally negligible. This is not much of a problem, since usually M_C will perform without errors. In the sequel, we don't use M_C so this issue can safely be ignored.

Definition 5.2.4 *Let $L_n(M; S)$ represent the average number of bits per symbol of source S that the $(t,k)-$encoding $M = (M_A, M_B, M_C)$ achieves. The parameter n is a scale value upon which the success probabilities of the algorithms and the length of their inputs depend.*

Since all the algorithms are probabilistic, for a fixed input the algorithm M_A may return different outputs. Let then $|M_A(n,x)|$ represent the expected length of $M_A(n,x)$ over all possible random values that M_A uses internally. Then,

$$L_n(M; S) = \frac{\sum_{x \in S_n^\lambda} p_n^\lambda(x) |M_A(n,x)|}{\lambda}.$$

Definition 5.2.5 *A $(t,k)-$entropy-sequence for S is a sequence $s_1, s_2, \ldots$ such that there exists a $(t,k)-$encoding scheme M for S with $L_n(M; S) = s_n$.*

The above definition is perhaps too general, and it might be a bit cumbersome. In particular, the parameter t could be forgotten by simply requiring that the scheme makes no mistakes neither in the decryption process nor in the breaking of the encoded text into blocks, for example, making M_B and M_C deterministic. Also, for $k = 0$, λ

becomes equal to 1 and $S_n^\lambda = S_n$, and in this case, we recover the more common definition of encoding over single symbols of a given source.

Definition 5.2.6 (Effective Entropy) *We say that a source S has effective entropy $H_c(S;n) \leq g(n)$ if there exists a $(t,k)-$entropy sequence $\langle s_n \rangle$ for S such that $s_n \leq g(n)$ for sufficiently large n.*

Similarly, $H_c(S;n) \geq g(n)$ if for every pair (t,k), every $(t,k)-$entropy sequence $\langle s_n \rangle$ for S satisfies $s_n \geq g(n)$.

This definition is very general, since it not only allows the encoder and the decoder to be probabilistic but it also considers the effect of patterns occurring in a longer text that would not be detected if the compression acted on each word individually. For this to be feasible, the definition considers the text has at most a number of words that is polynomial on the scale parameter n. This makes H_c hard to compare with the other notions of computational entropy, which are concerned with individual words. As such, $H_c(S;n)$ is to be compared with $H_\varepsilon^{\text{Yao}}(S_n^\lambda)$ and $E(K^t(S_n^\lambda))$. All these functions are based on inputs of λ times the length of a symbol of S, but H_c is scaled down by dividing by λ. To properly compare these measures, we consider instead the unscaled version of H_c, that is, $\lambda \cdot H_c(S;n)$.

5.2.2 BSW's Yao-type pseudoentropy ($H_\varepsilon^{\text{Yao}}$)

Definition 5.2.7 *Let $\mathcal{C}$ be a class capturing the notion of efficiently computable functions. Let S be a random variable representing the output of a random source. For any two functions $(c,d) \in \mathcal{C}$ defined as $c : [S] \to \{0,1\}^\ell$ and $d : \{0,1\}^\ell \to [S]$, let $D = \{x \in S : d(c(x)) = x\}$ and call this the codeset of c and d. Denote by $\mathcal{C}(\ell)$ the class of all such functions and by $\mathcal{D}(\ell)$ the class of all such D.*

Random variable S has computational ε-Yao-like entropy at least k, written $H_\varepsilon^{\text{Yao}}(S) \geq k$, if for all $\ell < k$ and all $D \in \mathcal{D}(\ell)$ it happens that $\Pr[S \in D] \leq 2^{\ell-k} + \varepsilon$ for some $\varepsilon > 0$.

This definition is often invoked in this chapter. Note that the class $\mathcal{C}$ is a parameter that can be instantiated by several classes. In this chapter we consider two of them: the class of functions computable in polynomial time, **FP**, in Theorems 5.3.3 and 5.3.5, and the class of functions computable in polynomial time with the help of randomness,

PPT, in Theorems 5.3.7 and 5.3.8. Note that these functions are not restricted to giving values in $\{0,1\}$. The reader should note that the notation for the previous entropies may be a bit misleading. Despite $H_\varepsilon^{\text{Yao}}$ having the word Yao as a parameter, this notion is due to Barak, Shaltiel and Wigderson. Yao's effective entropy is H_c.

5.3 Relations between H_c and $H_\varepsilon^{\text{Yao}}$

Given that these two definitions are analogues of entropy and min-entropy, it is interesting to find if the order relation between them still holds in the computational setting. That is, is it true that $H_\varepsilon^{\text{Yao}} \leq H_c$? This section investigates the answer to this question. We begin by giving a deterministic analog of H_c.

Definition 5.3.1 *$H_c^d(S;n)$ is defined as $H_c(S;n)$, with deterministic M_A and M_B.*

It is clear that if a deterministic encoding scheme achieves entropy $g(n)$ for some source S, then adding randomness to this scheme can only improve the compression, so

$$H_c(S;n) \leq H_c^d(S;n).$$

The following lemma will be needed in the sequel:

Lemma 5.3.2 *Given any language L over $\{0,1\}^{<n}$, it is possible to build another language L' isomorphic to this such that all words in L' have length n. Furthermore, the isomorphism can be computed and reverted in linear time.*

Proof: For any string $x \in L$, define $f\{0,1\}^{<n} \mapsto \{0,1\}^n : f(x) = 0^{n-|x|-1}1x$. It is easy to check that this construction can be made and reverted quickly. Now, it is necessary to show that all words thus formed are distinct. For any two words $x_1, x_2 \in L$ of distinct sizes, $f(x_1)$ and $f(x_2)$ have the leftmost 1 at different positions, so they are different. For any two distinct strings x_1, x_2 of the same size, the prefix appended to both strings is equal, so $f(x_1) \neq f(x_2)$ because $x_1 \neq x_2$. □

We give an outline of this section. First, Theorem 5.3.3 relates the deterministic version of H_c, Yao's effective entropy, with $H_\varepsilon^{\text{Yao}}$, the notion of min-entropy. It shows that if $H_\varepsilon^{\text{Yao}}(S_n^\lambda)$ is greater than $\lambda \cdot g(n) + i + 1$ for $i \geq 1$, then $H_c^d(S;n)$ will be lower bounded by a fraction of $g(n)$ that approaches 1 exponentially fast as i increases. However, if $i = 0$, this result is meaningless, for it simply implies that $H_c^d \geq 0$. To solve this case,

we employ a different technique in Theorem 5.3.5. The result is parallel to that of the previous theorem , showing that if $H_\varepsilon^{\text{Yao}}(S_n^\lambda) \geq \lambda \cdot g(n) + 1$, then $H_c^d(S;n)$ is at least greater than $g(n)$ minus a small factor. Corollaries 5.3.4 and 5.3.6 show that in each case H_c^d is greater than an affine function of $H_\varepsilon^{\text{Yao}}$. These theorems and corollaries can be used almost without changes for the randomized version of H_c instead, which establishes the relation sought in this section.

Theorem 5.3.3 *Consider the definition of Yao-type pseudoentropy (Def. 5.2.7) and instantiate the class $\mathcal{C}$ in that definition as* **FP**. *If there are a function $g : \mathbb{N} \mapsto \mathbb{R}^+$ and an integer $i > 0$ such that $H_\varepsilon^{\text{Yao}}(S_n^\lambda) \geq \lambda \cdot g(n) + (i+1)$ for sufficiently large n, then $H_c^d(S;n) \geq \left(1 - \frac{1}{2^i}\right) g(n)$.*

Proof: The proof is by the counter-positive. Suppose that $H_c^d(S;n) < \left(1 - \frac{1}{2^i}\right) g(n)$. Then, there is a (t,k)−encoding scheme $M = (M_A, M_B, M_C)$ that, for sufficiently large n, satisfies $L_n(M;S) < \left(1 - \frac{1}{2^i}\right) g(n)$. Then, for sufficiently large n, this condition is equivalent to

$$\sum_{x \in S_n^\lambda} p_n^\lambda(x) |M_A(n,x)| < \lambda \cdot \left(1 - \frac{1}{2^i}\right) g(n).$$

Define the set $D = \{x \in S_n^\lambda : |M_A(n,x)| \leq \lambda \cdot g(n)\}$. Then,

$$\begin{aligned} \lambda \cdot \left(1 - \frac{1}{2^i}\right) g(n) &> \sum_{x \in S_n^\lambda} p_n^\lambda(x) |M_A(n,x)| \geq \sum_{x \in S_n^\lambda \setminus D} p_n^\lambda(x) |M_A(n,x)| \\ &> \lambda \cdot g(n) \cdot \Pr[S_n^\lambda \notin D] \\ &\Rightarrow \Pr[S_n^\lambda \notin D] < \left(1 - \frac{1}{2^i}\right). \end{aligned}$$

This means that $\Pr[S_n^\lambda \in D] > \frac{1}{2^i}$. To show that $H_\varepsilon^{\text{Yao}}(S_n^\lambda) < \lambda \cdot g(n) + i + 1$, it suffices to give a pair of functions $c', d' \in \mathcal{C}(\lambda \cdot g(n) + 1)$ and respective codeset D' such that $\Pr[S_n^\lambda \in D'] > \frac{2^{\lambda \cdot g(n)+1}}{2^{\lambda \cdot g(n)+i+1}} + \varepsilon = \frac{1}{2^i} + \varepsilon$. Since every $x \in D$ can be compressed to a string of length at most $\lambda \cdot g(n)$, by Lemma 5.3.2 we can efficiently compute a set D' that is isomorphic to D and such that every element $y \in D'$ has length $\lambda \cdot g(n) + 1$. The reverse operation is also efficient. Let f be this isomorphism. Then, let $c'(x) = f(M_A(n,x))$ and $d'(y) = M_B(f^{-1}(y))$. Since $\Pr[S_n^\lambda \in D] = \Pr[S_n^\lambda \in D']$, the theorem follows. □

Corollary 5.3.4 *Let $g(n)$ be some function from $\mathbb{N}$ to $\mathbb{R}^+$ such that $H_\varepsilon^{\text{Yao}}(S_n^\lambda) = \lambda \cdot g(n) + (i+1)$ for some integer $i > 0$. Then, $\lambda \cdot H_c^d(S;n) \geq \left(1 - \frac{1}{2^i}\right)\left(H_\varepsilon^{\text{Yao}}(S_n^\lambda) - i - 1\right)$.*

Proof: By Theorem 5.3.3, $H_c^d(S;n) \geq \left(1-\frac{1}{2^i}\right) g(n) \Leftrightarrow \lambda \cdot H_c^d(S;n) \geq \left(1-\frac{1}{2^i}\right) \cdot \lambda \cdot g(n) = \left(1-\frac{1}{2^i}\right) \cdot (H_\varepsilon^{\text{Yao}}(S_n^\lambda) - i - 1)$. □

If $i = 0$, the last theorem does not say much, since it simply implies that in this case $H_c^d \geq 0$. We can get a more meaningful bound with a different technique:

Theorem 5.3.5 *Consider the definition of Yao-type pseudoentropy (Def. 5.2.7) and instantiate the class $\mathcal{C}$ in that definition as* **FP**. *If there is a function $g : \mathbb{N} \mapsto \mathbb{R}^+$ such that $H_\varepsilon^{\text{Yao}}(S_n^\lambda) \geq \lambda \cdot g(n) + 1$ for sufficiently large n, then $H_c^d(S;n) \geq g(n) - 1/\lambda$.*

Proof: Fix any efficient deterministic encoding scheme $M = (M_A, M_B, M_C)$. Let $c(x)$ be the restriction of $M_A(n,x)$ to the domain $R = \{x \in S_n^\lambda : |M_A(n,x)| < \lambda \cdot g(n)\}$. Then, $d(x) = M_B(x)$. Using Lemma 5.3.2, there are functions c', d' and $D = \{x \in S_n^\lambda : d'(c'(x)) = x\}$ such that all and only elements in R are in D and $|c'(x)| = \lambda \cdot g(n)$ for all $x \in R$. Then, by assumption, $\Pr[X \in D] \leq \frac{2^{\lambda \cdot g(n)}}{2^{\lambda \cdot g(n)+1}} = 1/2 \Leftrightarrow \Pr[|M_A(n,x)| \geq \lambda \cdot g(n)] \geq 1/2$, and $L_n(M;S)$ can be estimated:

$$\begin{aligned} \lambda \cdot L_n(M;S) &= \sum_{x \in S_n^\lambda} p(x)|M_A(n,x)| \\ &\geq \sum_{x \in R} p(x)|M_A(n,x)| + 1/2 \cdot \lambda \cdot g(n). \end{aligned} \tag{5.1}$$

The objective of the proof now is to show a lower bound for $L_n(M;S)$. The worst case happens when the $x \in S_n^\lambda$ that have shortest $|M_A(n,x)|$ descriptions have maximal probability. The first half of (5.1) is studied next.

For any $\ell < \lambda \cdot g(n)$, let $M_A(n,x)|_\ell$ be the restriction of M_A to those strings x that are transformed into strings of length ℓ. This new restriction is also an efficient coding for the strings in its domain. The result for other strings is not defined, for example we can assume they are all transformed into the empty string. Then we have

$$\Pr[|M_A(n,x)| = \ell] \leq 2^l / 2^{\lambda \cdot g(n)+1} \tag{5.2}$$

$$\Pr[|M_A(n,x)| < \ell] \leq 2^l / 2^{\lambda \cdot g(n)+1}, \tag{5.3}$$

the first by the assumption on $H_\varepsilon^{\text{Yao}}$ and the second by the same assumption after using Lemma 5.3.2. We can write $\sum_{x \in R} p(x)|M_A(n,x)| = \sum_{i=0}^{\lambda \cdot g(n)-1} i \cdot \Pr[|M_A(n,x)| = i]$.

To find a lower bound to this sum, we must attribute maximal probability to the lowest values of i. Respecting bounds (5.2) and (5.3), we find that the maximal value for each $\Pr[|M_A(n,x)| = i]$ must be $2^i / 2^{\lambda \cdot g(n)+1}$. Then, $\Pr[|M_A(n,x)| < \lambda \cdot g(n)] = \frac{1}{2} - \frac{1}{2^{\lambda \cdot g(n)+1}}$

which is as close to the upper bound $\frac{1}{2}$ as possible while respecting the assumption on $H_\varepsilon^{\text{Yao}}$. Now,

$$\begin{aligned}\sum_{x\in R} p(x)|M_A(n,x)| &= \sum_{i=0}^{\lambda\cdot g(n)-1} i\cdot \Pr[|M_A(n,x)|=i] \\ &= \sum_{i=0}^{\lambda\cdot g(n)-1} i\cdot \frac{2^i}{2^{\lambda\cdot g(n)+1}} \\ &= \frac{1}{2^{\lambda\cdot g(n)+1}}\cdot\left[2^{\lambda\cdot g(n)}\cdot(\lambda\cdot g(n)-2)+2\right] \geq \frac{\lambda\cdot g(n)-2}{2}.\end{aligned}$$

But then, $\lambda\cdot L_n(M;S) \geq \lambda\cdot g(n)-1$. Since this must happen for any encoding scheme, it follows that $\lambda\cdot H_c^d(S;n) \geq \lambda\cdot g(n)-1 \Rightarrow H_c^d(S;n) \geq g(n)-1/\lambda$. □

Corollary 5.3.6 *Let $g(n)$ be some function from $\mathbb{N}$ to $\mathbb{R}^+$ such that $H_\varepsilon^{\text{Yao}}(S_n^\lambda) = \lambda\cdot g(n)+1$. Then, $H_c^d(S;n) \geq H_\varepsilon^{\text{Yao}}(S_n^\lambda)-2$.*

Proof: By Theorem 5.3.5, $\lambda\cdot H_c^d(S;n) \geq \lambda\cdot g(n)-1 = H_\varepsilon^{\text{Yao}}(S_n^\lambda)-2$. □

The previous results concern the deterministic version of H_c. That is because we have made $\mathcal{C} = \mathbf{FP}$. It is commonly accepted today that the class of probabilistic polynomial time algorithms (**PPT**) may be a better representation of real-world efficiency. Letting $\mathcal{C}$ equal to that class, the previous results all hold for H_c using essentially the same proofs.

Theorem 5.3.7 *Consider the definition of Yao-type pseudoentropy (Def. 5.2.7) and instantiate the class $\mathcal{C}$ in that definition as* **PPT**. *If there are a function $g:\mathbb{N}\mapsto\mathbb{R}^+$ and an integer $i>0$ such that $H_\varepsilon^{\text{Yao}}(S_n^\lambda) \geq \lambda\cdot g(n)+(i+1)$ for sufficiently large n, then $H_c(S;n) \geq \left(1-\frac{1}{2^i}\right) g(n)$.*

Theorem 5.3.8 *Consider the definition of Yao-type pseudoentropy (Def. 5.2.7) and instantiate the class $\mathcal{C}$ in that definition as* **PPT**. *If there is a function $g:\mathbb{N}\mapsto\mathbb{R}^+$ such that $H_\varepsilon^{\text{Yao}}(S_n^\lambda) \geq \lambda\cdot g(n)+1$ for sufficiently large n, then $H_c(S;n) \geq g(n)-1/\lambda$.*

The corresponding corollaries also hold. These theorems establish the expected relation, that up to a multiplicative constant, the effective entropy is at least as large as Yao-type pseudoentropy.

5.4 Relations between H_c and K^t

By definition, Yao's effective entropy identifies the probabilistic polynomial encoding scheme that achieves the best compression. This section shows that despite the randomness available to the encoding schemes, the average time-bounded Kolmogorov complexity is still a lower bound for H_c. Since K^t is defined for deterministic Turing Machines, we first compare K^t with the deterministic version of H_c and then address the general case.

Theorem 5.4.1 $\lambda \cdot H_c^d(S;n) \geq E(K^t(S_n^\lambda)) - O(1)$, *for some polynomial* $t(\cdot)$.

Proof: Let $H_c^d(S;n) = g(n)$ and $M = (M_A, M_B, M_C)$ be any $(\tau, k)-$encoding scheme for S that achieves $L_n(M;S) = g(n)$, where τ is some positive constant[1]. Then every $x \in S$ can be computed by M_B and a codeword output by M_A. Furthermore, these algorithms all run in time polynomial on n, say $q(n)$. Since M_B is fixed, its size is a constant and therefore so is its shortest description. If we input this to a universal Turing machine, this is able to decode B's description, say in time $s(|M_B|)$, and simulate it with at most a polynomial increase in its running time. Fix a polynomial $t(n)$ that upper bounds the running time of the simulation of B. Then, for all $x \in S_n^\lambda$, $K^t(x) \leq |M_A(n,x)| + K^s(M_B) = |M_A(n,x)| + O(1)$.

But then,

$$\begin{aligned} E(K^t(S_n^\lambda)) &= \sum_{x \in S_n^\lambda} p_n^\lambda(x) K^t(x) \\ &\leq \sum_{x \in S_n^\lambda} p_n^\lambda(x) |M_A(n,x)| + O(1) \\ &= \lambda \cdot L_n(M;S) + O(1) \\ &= \lambda \cdot H_c^d(S;n) + O(1). \end{aligned}$$

Since for all large enough n such an M exists, the theorem follows. □

To allow for probabilistic machines, we consider a program for x that has access to a quick-pseudorandom generator (PRG), according to the terminology of Nisan and Wigderson ([NW94]). The existence of such a PRG is equivalent to the existence of a hard function in **EXP**. Recall that the class **EXP** contains all languages that can be decided within time 2^{n^c}, where n stands for the length of the output and c is any positive integer.

[1] Recall that this defines the probabilities of error in the operation of algorithms M_A, M_B or M_C.

Theorem 5.4.2 *Fix a $(\tau, k)-$encoding scheme $(M_A; M_B; M_C)$ for S as guaranteed by the definition of $H_c(S; n)$, with $\tau \geq 2$ and let $q(n) \geq \sqrt{2n}$ be a polynomial upper bounding the execution time of these algorithms on input $(n; \cdot)$.*

If there is a function $f : \{0,1\}^* \mapsto \{0,1\}$ in **EXP** *such that for some constants $k, c > 0$ and large enough n with $n^{c/2} \geq q(n)$, for all circuits C_n of size at most n^c, $\Pr[C(x) \neq f(x)] > (n^c)^{-k}$, then $\lambda \cdot H_c(S; n) \geq E(K^t(S_n^\lambda)) - O(1)$, for some polynomial $t(\cdot)$ (to be defined in the proof).*

Proof: Since M_B is a probabilistic algorithm that runs in time $q(n)$, there is a probabilistic circuit C_B of size at most $q(n)^2$ that computes exactly like M_B using at most $q(n)$ random bits.

By the construction of [NW94], the existence of f implies that there is a function $G : \log n \mapsto q(n)^2$ such that for any circuit C_n of size at most $q(n)^2$, $|\Pr_{r \in \{0,1\}^{q(n)^2}}[C_n(r) = 1] - \Pr_{z \in \{0,1\}^{\log n}}[C_n(G(z)) = 1]| < 1/q(n)^2$.

Let $p_{M_B}(y)$ be the probability that $M_B(n; y)$ does not output the correct source element for the codeword y. By definition, $p_{M_B} < 1/n^\tau$ and by construction this is the probability that at least one bit in the output of C_B is wrong. Let C_B^i be a circuit that returns the i^{th} bit of C_B. This C_B^i is of the same size as C_B and returns a boolean function. By hard-wiring y in it, we get a circuit $C_{B,y}^i$ of size $q(n)^2$ that takes a $q(n)-$bit input (by ignoring the rest of the random tape) corresponding to the random bits, and by the definition of G, $|\Pr_{r \in \{0,1\}^{q(n)}}[C_{B,y}^i(r) = 1] - \Pr_{z \in \{0,1\}^{\log n}}[C_{B,y}^i(G(z)) = 1]| < 1/q(n)^2$. This implies that the probability that each $C_{B,y}^i$ is wrong when using pseudoramdoness is at most $1/n^\tau + 1/q(n)^2$. By the union bound, the probability that C_B, and therefore M_B, outputs a wrong element when given $q(n)$ pseudorandom bits output by G is at most $n \cdot (1/n^\tau + 1/q(n)^2) \leq 1/n + 1/2$. This means there is at least one string in the range of G that succeeds.

Let this string be w. Then, by computing $M_B(n; M_A(n, x))$ with randomness $G(w)$, we produce x. We can give a program for x accordingly, by joining all the pieces. The generator G might be too big to give explicitly, since it depends on n, so we produce it on the fly, instead. We give the code of M_B explicitly as $|M_B|$, because for n large enough M_B should be considered a constant. Finally, we have to give the random seed w and the output of the encoder. We leave the latter for the end of the program so that it does not need to be self-delimited. The seed, however, must be encoded so that the program can separate it from its surroundings, and we achieve it by using a standard encoding that doubles the number of bits used. Both G can be produced

and $G(w)$ be computed in time $O(poly(n))$. Let $p(n)$ be a polynomial upper bounding these times. Then,

$$K^t(x) \leq |M_A(n,x)| + 2|w| + K^p(G) + |M_B| + O(1) \Rightarrow$$
$$K^t(x) \leq |M_A(n,x)| + 2\log n + O(1)$$

where $t(n)$ is a polynomial upper bounding the time necessary to produce G, execute $G(w)$ and run M_B, for a maximum of $2p(n) + q(n)$, and we lumped $K^p(G)$ and $|M_B|$ in one constant term.

Therefore, we get $\lambda \cdot H_c(S;n) \geq E(K^t(S_n^\lambda)) - O(\log n)$. □

5.5 Relation between $H_\varepsilon^{\text{Yao}}$ and K^t

The objective of this section is to analyse whether a meaningful relationship can be found between Yao-type pseudoentropy and the average value of $K^t(X)$. To begin, we give a relaxation of $H_\varepsilon^{\text{Yao}}$ that can be compared to $E(K^t(X))$.

Definition 5.5.1 *Consider the definition of Yao-like pseudoentropy (Def. 5.2.7). Let c, d be a pair of computable functions as before, but only d is required to belong to $\mathcal{C}$. Let $\mathcal{D}^+$ be the corresponding class of codesets.*

A random variable X has inefficient computational Yao-type pseudoentropy at least k, written $H_\varepsilon^{\text{Yao}+}(X) \geq k$, if $\Pr[X \in D] \leq 2^{\ell-k} + \varepsilon$ for all $\ell < k$ and all $D \in \mathcal{D}^+(\ell)$.

This definition includes at least all the functions allowed by the definition of $H_\varepsilon^{\text{Yao}}(X)$. Thus if a certain property holds for all the functions allowed by Definition 5.5.1, then it also holds for all the functions allowed by Definition 5.2.7. Hence the following lemma:

Lemma 5.5.2 *For any random variable X, the following holds: $H_\varepsilon^{\text{Yao}+}(X) \leq H_\varepsilon^{\text{Yao}}(X)$.*

The relation between $H_\varepsilon^{\text{Yao}+}$ and $E(K^t(X))$ can be shown by a theorem that uses the same technique of Theorem 5.3.3.

Theorem 5.5.3 *Consider the class $\mathcal{C}$ referred in Definition 5.5.1 and let $\mathcal{C} = \mathbf{FP}$. Then, for any integer $i > 0$, if $H_\varepsilon^{\text{Yao}+}(X) \geq k+i$ it follows that $E(K^t(X)) \geq \left(1 - \frac{1}{2^i}\right) k$ for some polynomial $t(\cdot)$.*

Proof: Let $c(x)$ be a function that returns the shortest program for x that runs in time $t(|x|)$, i.e., $|c(x)| = K^t(x)$. Let d be the corresponding decoding function $d(y)$, which simply consists of executing y in the reference universal Turing machine. Define $D = \{x \in X : |c(x)| < k\}$ and let $c'(x)$ be a function that takes $c(x)$ and expands the result to exactly k bits by Lemma 5.3.2. The respective decoder function $d'(y)$ first recovers $c(x)$ from y and then executes d on the result.

We prove the counter-positive of the theorem statement. Suppose that $E(K^t(X)) < \left(1 - \frac{1}{2^i}\right) k$. Then, $\left(1 - \frac{1}{2^i}\right) k > \sum_{x \in X} p(x) K^t(x) \geq \sum_{x \in X \setminus D} p(x)|c(x)| \geq k \cdot \Pr[X \notin D]$ so

$$\Pr[X \in D'] = \Pr[X \in D] > \frac{1}{2^i} = \frac{2^k}{2^{k+i}}. \tag{5.4}$$

The function d is efficient, because it only has to run a universal Turing machine for a polynomial number of steps. Then, d' is also efficient, which implies, together with (5.4), that $H_\varepsilon^{\text{Yao+}}(X) < k + i$. □

If the inverse implication were true, that is, if a large expected value of Kolmogorov complexity implied a large value of $H_\varepsilon^{\text{Yao+}}(X)$, then this would imply a lower bound to $H_\varepsilon^{\text{Yao}}(X)$ and we'd be able to relate $E(K^t(X))$ to $H_\varepsilon^{\text{Yao}}(X)$. Unfortunately, that is not the case.

Example 2 *Consider a set of binary strings of length at most $n > k$. Let b be the length of the shortest program accepted by the reference Turing machine. Let X be a random distribution over this set with the following associated probability function, where $t(\cdot)$ is some fixed polynomial:*

$$\Pr[K^t(x) = b] = \frac{2^{b+1} + 1}{2^k},$$
$$\Pr[K^t(x) \in \{0, 1, \ldots, b-1, b+1, \ldots, k-1\}] = 0,$$
$$\Pr[K^t(x) \geq k] = 1 - \frac{2^{b+1} + 1}{2^k},$$

for some large enough k.

We give a randomized algorithm that outputs samples according to this distribution. It generates a random number between 0 and 1 and if it is at most $(2^{b+1} + 1)/2^k$ it runs the shortest program accepted by the reference Turing machine for $t(n)$ steps, outputting whatever this program produces. If not, then the program produces some string y of length n at random and then computes $K^t(y)$. If this is greater than k, the program outputs y, otherwise it generates a new string and repeats the process. The

theorem of incompressibility guarantees that there are strings with complexity at least k and that the probability of generating one at random increases exponentially as k gets smaller than n.

Consider the encoding scheme that transforms each string x in its shortest program, and let $\ell = b$. The corresponding codeset D has only one string, but $\Pr[x \in D] = (2^{b+1}+1)/2^k > 2^\ell/2^{k-1}$ *so* $H_\varepsilon^{\text{Yao}+}(X) < k-1$. *However,* $E(K^t(X)) \geq k \cdot \left(1 - \frac{2^{b+1}+1}{2^k}\right) + b \cdot \left(\frac{2^{b+1}+1}{2^k}\right) \geq k - \frac{k \cdot (2^{b+1}+1)}{2^k}$, *which for $k \geq 2^{b+1}+1$ is greater than $k-1$.*[2]

This process can be generalized for the following distribution

$$\begin{aligned}
&\Pr[K^t(x) = b] = \frac{2^{b+1}+1}{2^{k-\alpha+1}}, \\
&\Pr[K^t(x) \in \{0, 1, \ldots, b-1, b+1, \ldots, k-1\}] = 0, \\
&\Pr[K^t(x) \geq \beta \cdot k] = 1 - \frac{2^{b+1}+1}{2^{k-\alpha+1}},
\end{aligned}$$

for appropriate k. Then, $H_\varepsilon^{\text{Yao}+}(X) < k - \alpha$ *but* $E(K^t(X)) \geq \beta k \cdot \left(1 - \frac{2^{b+1}+1}{2^{k-\alpha+1}}\right)$, *which for $\beta \geq 1$ grows to be arbitrarily larger than $k - \alpha$.*

The results in the previous sections can be summed up in the following inequalities, up to the approximations shown in the Theorems:

$$\begin{aligned}
H_\varepsilon^{\text{Yao}+}(X^\lambda) &\leq H_\varepsilon^{\text{Yao}}(X^\lambda) \leq \lambda \cdot H_c(X) \\
H_\varepsilon^{\text{Yao}+}(X^\lambda) &\leq E(K^t(X^\lambda)) \leq \lambda \cdot H_c(X).
\end{aligned}$$

It remains an open question to find the relation between $E(K^t(X))$ and $H_\varepsilon^{\text{Yao}}(X)$. As shown later, for the universal distribution $\mathbf{m}^t$, $E(K^t(X))$ is markedly superior to $H_\varepsilon^{\text{Yao}}(X)$, but there can be other distributions for which the relation is inverted.

[2]This estimate is overly conservative. The least k for which the former fraction is over $k-1$ can be found with the help of Lambert's W function (see [CGH+96]) to be the first integer above $\frac{-W_{-1}\left(\frac{-\ln 2}{2^{b+1}+1}\right)}{\ln 2}$. Since $-W_{-1}(-x) + \ln -W_{-1}(-x) = \ln x$, and for $x > 0$ the term $\ln x$ is always less than $x/2$, we have for values of x in $(0, 1/e]$ that $0 \leq -W_{-1}(-x) \leq -2 \ln x$. Then, the fraction above is upper bounded by $\frac{-2 \ln \frac{\ln 2}{(2^{b+1}+1)}}{\ln 2} < 2b + 4$ for $b \geq 1$, after some manipulation. So, the least k for which the example works is at most $2b + 4$. We note the logarithms here are natural logarithms instead of using base 2.

5.6 Relations for Specific Distributions

This section analyses the uniform and the $\mathbf{m}^t$ distributions. Throughout this section, U_n denotes the uniform distribution over binary strings of length n.

5.6.1 Uniform Distribution

Theorem 5.6.1 *Let X be a uniformly distributed random variable over the set $\{0,1\}^n$. Then, $H_\varepsilon^{\text{Yao}}(X) = n$.*

Proof: First we show that $H_\varepsilon^{\text{Yao}}(X) \geq n$. In fact, for every pair of functions $c, d \in \mathcal{C}(\ell)$ the set $D = \{x \in X : c(d(x)) = x\}$ has at most 2^ℓ different words. Since all of them have probability equal to 2^{-n}, we get $\Pr[X \in D] = |D|/2^n \leq 2^\ell/2^n$. Then by definition, $H_\varepsilon^{\text{Yao}}(X) \geq n$.

To prove the converse, we have to show that there is some $\ell < n+1$ and a pair of functions $c, d \in \mathcal{C}(\ell)$ such that for $D = \{x \in X : c(d(x)) = x\}$, $\Pr[X \in D] > 2^\ell/2^{n+1}$. There are exactly 2^n strings in this distribution, all of them with length equal to n. Then we can code them by the identity function. Therefore, $|D| = 2^n$ and $\Pr[X \in D] = 1 > 2^n/2^{n+1}$. Then, $H_\varepsilon^{\text{Yao}}(X) < n+1$. □

Next we prove a similar result for H_c. First, we state the following lemma, derived from the fact that $K^t(s|n) \leq n + O(1)$, for any $s \in \{0,1\}^n$.

Lemma 5.6.2 *For any random variable S over binary strings of length at most n, $E(K^t(S|n)) \leq n + O(1)$.*

Theorem 5.6.3 *Let X be a uniformly distributed random variable over $\{0,1\}^n$. Then, $E_{X \sim U_n}(K^t(X|n)) = \Theta(n)$.*

Proof: Lemma 5.6.2 shows $E_{X \sim U_n}(K^t(X|n)) \leq n + O(1)$. Now we prove the converse by giving a lower bound for $E_{X \sim U_n}(K^t(X|n))$.

It is known that $K^t(x|n) \leq n + O(1)$, where $n = |x|$, so we can write $E(K^t(U_n)) = \frac{1}{2^n} \sum_{i=0}^{n+O(1)} i \cdot f(i)$ where $f(i)$ is the number of strings x with $K^t(x|n) = i$. This sum is minimum when the complexities $K^t(x|n)$ are minimum, this is, when we exhaust the programs of short length first. This is done by considering that a string has complexity ℓ only if there are distinct strings using up each of the programs of length less than

ℓ. We consider the worst case where the set of programs is not prefix-free and so all programs of each size can be used. In this scenario, there are 2^i programs of size i. Thus, by Lemma 2.3.1

$$E_{X\sim U_n}(K^t(X|n)) \geq \frac{1}{2^n}\left(n + \sum_{i=0}^{n-1} i \cdot 2^i\right)$$
$$\geq \frac{1}{2^n} \cdot (2^n \cdot (n-2) + 2) \geq n - 2.$$

□

The next corollary follows from Theorems 5.6.3 and 5.4.2.

Corollary 5.6.4 *Let X be a uniformly distributed random variable over $\{0,1\}^n$. If there are hard functions in* **EXP**, *then $H_c(X;n) = \Theta(n)$.*

The previous theorems show that for the uniform distribution, all three notions are asymptotically equivalent.

5.6.2 Universal Distribution $\mathbf{m}^t$

This section analyses the relation between these notions under the universal distribution $\mathbf{m}^t$.

Definition 5.6.5 *The universal distribution $\mathbf{m}^t$ is defined as $\mathbf{m}^t(x) = 2^{-K^t(x)}$ (see [LV97], pg 506).*

This distribution is computable in time $t(n)2^{n+1}$ for $n = |x|$. In the remainder of this section, we consider only the restriction to binary strings of length n, $\mathbf{m}^t_n(x)$. Unlike the case for the uniform distribution, there is a noticeable difference between $H^{\text{Yao}}_\varepsilon(X)$ and $H_c(X;n)$ when X is distributed according to $\mathbf{m}^t_n$.

Theorem 5.6.6 *Let X be a random variable over $\{0,1\}^n$ with $\Pr[X = x] = \mathbf{m}^t_n(x)$ for some polynomial $t(\cdot)$. Then, for any constant $c' > 0$, $H^{\text{Yao}}_\varepsilon(X) < 2c' \log n + 1$.*

Proof: Let $b = 2c' \log n + 1$. We find some ℓ, and a pair of functions $f, g \in \mathcal{C}(\ell)$ such that for $D = \{x \in X : g(f(x)) = x\}$, it happens that $\Pr[X \in D] > 2^\ell/2^b + \varepsilon$.

Consider the following algorithm. When f receives $x \in \{0,1\}^n$, it executes some universal prefix-free Turing machine U with all programs p_i of length up to $c' \log n$ in parallel for at most $t(n)$ steps. If there is at least some i such that $U^t(p_i) = x$, then $f(x)$ is the shortest such program. Otherwise, it returns the empty string. This program runs in time polynomial in n and returns some shortest prefix-free program for x that runs in time $t(n)$. Therefore, $|p_i| = K^t(x)$. The set D associated with this function contains only and all strings x of size n with $K^t(x) \leq c' \log n$. Since all these strings form a prefix-free code, they can be padded with zeroes so that they all have length $c' \log n$.

There certainly is at least one string in this set, for example, the string x_0 composed of n zeroes has $K^t(x_0) \leq \log n + O(1)$. Therefore, $|D| > 0$. For all $x \in D$, $K^t(x) \leq c' \log n \Leftrightarrow \mathbf{m}_n^t(x) \geq 1/n^{c'}$. Then $\Pr[X \in D] \geq |D| \ / \ n^{c'} \geq 1 \ / \ n^{c'}$. Let $\ell = c' \log n$. Then, $\Pr[X \in D] \geq 1/2^\ell$. Since $b = 2\ell + 1$, it follows that $1/2^\ell > 2^\ell/2^b$. Therefore, $H_\varepsilon^{\text{Yao}}(X) < 2c' \log n + 1$.

□

It is known that $\mathbf{m}^{t'}(X)$ dominates all distributions $P(X)$ computable in time $t(n)$, where $t'(n) = nt(n)$ and $t(n)$ is a polynomial, as stated in Theorem 2.9.27. This can be used to prove the next theorem:

Theorem 5.6.7 *If there is a $t-$time computable distribution X with probability density function $P(x)$ for some polynomial $t(n)$ such that $H_\varepsilon^{\text{Yao}}(X) < f(n)$, then $H_\varepsilon^{\text{Yao}}(Y) < f(n) + K^{t'}(P) + O(1)$, where $t(\cdot)$ is a polynomial, $t'(n) = nt(n)$ and Y is a random variable with the same support of X but with $\Pr[Y = x] = \mathbf{m}_n^{t'}(x)$.*

Proof: By definition, there are functions $c : X \to \{0,1\}^\ell$ and $d : \{0,1\}^\ell \to X$ for some $\ell < f(n)$ such that for $D = \{x \in X : d(c(x)) = x\}$, $\sum_{x \in D} P(x) > \frac{2^\ell}{2^{f(n)}} + \varepsilon$.

Since P is $t-$time computable, then $P(x) \leq \mathbf{m}^{t'}(x) \cdot 2^{c_P}$, for $c_P = K^{t'}(P) + O(1)$ as in Theorem 2.9.27. Therefore, $\sum_{x \in D} \mathbf{m}^{t'}(x) > \frac{1}{2^{c_P}} \sum_{x \in D} P(x) \geq \frac{2^\ell}{2^{f(n)+c_P}}$. Then, $H_\varepsilon^{\text{Yao}}(Y) < f(n) + c_P = f(n) + K^{t'}(P) + O(1)$. □

It is possible to prove a result for H_c opposite to Theorem 5.6.6 that is a consequence of the following Theorem:

Theorem 5.6.8 *For the universal distribution $\mathbf{m}^t$, $E_{X \sim \mathbf{m}_n^t}(K^t(X|n)) = \Theta(n)$.*

Proof: By Lemma 5.6.2 we have $E_{X \sim \mathbf{m}_n^t}(K^t(X|n)) \leq n + O(1)$.

Consider the minimal prefix-free programs for all strings of length n. It is known that $K^t(x|n) \leq n + O(1)$. Let δ be the constant represented by $O(1)$. We can write $E_{X\sim\mathbf{m}_n^t}(K^t(X|n)) = \sum_{i=0}^{n+\delta} \frac{i}{2^i} \cdot f(i)$ where $f(i)$ is the number of strings that have $K^t(x) = i$. Since 2^i is the dominant term in the fraction, the minimum sum is achieved when all the strings have complexity as large as possible. There are 2^n strings of length n, so assume the worst case $f(i) = 2^n$ for $i = n + \delta$ and 0 everywhere else.

The total probability considering this arrangement is $\frac{1}{2^\delta} < 1$, which means that actually there must be at least one string with complexity lower than $n + \delta$ to compensate for the missing probability. This provokes an increase in the sum, so this arrangement gives a lower bound to $E_{X\sim\mathbf{m}_n^t}(K^t(X|n))$:

$$E_{X\sim\mathbf{m}_n^t}(K^t(X|n)) \geq 2^n/2^{n+\delta} \cdot (n + \delta) = \Omega(n).$$

□

Corollary 5.6.9 *Let $t(\cdot)$ be some polynomial and X be a random variable over $\{0,1\}^n$ with $\Pr[X = x] = \mathbf{m}_n^t(x)$. If there are hard functions in* **EXP**, *then $H_c(X; n) = \Theta(n)$.*

The same idea of the proof of Theorem 5.6.6 can be used to show the separation between $H_c(X; n)$ and metric type pseudoentropy for X distributed according to $\mathbf{m}_n^t$. We first give the definition presented in Lemma 3.3 of [BSW03] and then list the corresponding theorem.

Definition 5.6.10 *Let X be a random variable over a set S. For every class $\mathcal{C}$ which is closed under complement and for every $k \leq \log|S| - 1$ and ε, $H_\varepsilon^{\text{Metric}}(X) \geq k$ if and only if for every set D whose characteristic function belongs to $\mathcal{C}$, $\Pr[X \in D] \leq \frac{|D|}{2^k} + \varepsilon$.*

Theorem 5.6.11 *For any constant $c' > 0$, and some polynomial $t(\cdot)$, and a random variable X such that $\Pr[X = x] = \mathbf{m}_n^t(x)$, $H_\varepsilon^{\text{Metric}}(X) < c' \log n + 1$.*

Proof: The proof is similar to the one for Theorem 5.6.6, and it even uses some constructions from it.

Let $\mathcal{C} = \mathbf{P}$ and $\mathcal{C}(\ell)$ be the restriction of this class as outlined in the definition of Yao-type pseudoentropy (Def. 5.2.7). Now, consider the pair of functions $c, d \in \mathcal{C}(\ell)$ and the corresponding codeset D given in that proof.

We build a predicate $A \in \mathcal{C}$ such that $\{x : A(x) = 1\} = D$. For that, let $A(x)$ evaluate $d(c(x))$ and output 1 if and only if the result is x. Since both c and d are efficient and deterministic, so is $A(x)$. Then, all and only elements in D satisfy A. As was seen in the proof of Theorem 5.6.6, $\Pr[X \in D] \geq |D|/n^{c'}$. Plugging $c' \log n$ for k in Definition 5.6.10, we get that $H_\varepsilon^{\text{Metric}}(X) \leq c' \log n$.

□

We give a result for H_c parallel to that of Theorem 5.6.7.

Theorem 5.6.12 *Suppose there is a polynomial-time computable distribution X such that $H_c(X; n) \geq g(n)$. Let $t(\cdot)$ be a polynomial and Y be another random variable with the same support as X but such that $\Pr[Y = x] = \mathbf{m}_n^t(x)$. Then $H_c(Y; n) \geq g(n)/2^{c_P}$, for $c_P = K^{t'}(P) + O(1)$ and $t'(n) = nt(n)$.*

Proof: For any (t, k)−encoding scheme $M = (M_A, M_B, M_C)$ for X, we have, by definition, that $L_n(M; X) \geq g(n) \Leftrightarrow \frac{\sum_x p_n^\lambda(x)|M_A(x)|}{\lambda} \geq g(n)$. Recall that $x = \langle x_1, \ldots, x_\lambda \rangle$ is a word distributed according to X^λ. Since X is polynomial-time computable, then for $c_P = K^{t'}(P) + O(1)$, $\mathbf{m}_n^{t'}(x_i) \geq p_n(x_i)/2^{c_P} \Leftrightarrow K^{t'}(x_i) \leq \log 1/p_n(x_i) + c_P$. The proof of this theorem in [LV97] uses the description of P to reconstruct the distribution and thence the string that is sought.

This reasoning can also be applied to x as a whole, but now the reconstruction of P need be done only once. Thus, c_P is not multiplied by λ. Therefore,

$$K^{t'}(x) \leq \log 1/p_n^\lambda(x) + c_P \Leftrightarrow \mathbf{m}_n^{t'}(x) \geq p_n^\lambda(x)/2^{c_P}$$

and so

$$L_n(M; Y) \geq \frac{\sum_x \frac{p_n^\lambda(x)}{2^{c_P}} |M_A(x)|}{\lambda} \geq \frac{g(n)}{2^{c_P}}$$

and the theorem follows. □

The results in this section show that there is a computable distribution that clearly separates H_c from $H_\varepsilon^{\text{Yao}}$ and especially $H_\varepsilon^{\text{Metric}}$. Since $H_\varepsilon^{\text{Metric}}$ is equivalent to HILL-type pseudoentropy, this establishes a separation between pseudoentropy and the maximum compressibility of a samplable source. This result is similar to Wee's, which is relative to an oracle, and depends on the existence of good PRGs. This assumption is needed only because the coding function may be a probabilistic algorithm. If we allow only deterministic encodings, the result follows unconditionally.

It seems reasonable to believe that the average of bounded Kolmogorov Complexity is a strict lower-bound for the maximum compressibility of a source. Yao's effective entropy cannot be easily computed. We'd have to consider all possible encoding schemes and run them over all strings to compare their averages. The problem is that the number of possible schemes is infinite. However, Kolmogorov Complexity theory gives us a framework for computing this value, if one can spend a large amount of time: we can use the standard trick of, for every string in the support of S_n^λ, enumerating all programs of size up to $\lambda \cdot n$ in dovetailing fashion, from the shortest to the longest, and running each of them only up to a fixed number of steps polynomial on n. As soon as the shortest program is found for x, keep its size and advance for the next string. This is a finite computation, since the number of strings, of programs and execution time are all finite quantities. Although it is not our aim in this thesis to give support to some measure or other of computational entropy or maximum compressibility, we think the above favours the expected value of time-bounded Kolmogorov complexity as a good candidate for the true measure of maximum compressibility of a random source.

Chapter 6

Conclusion and Open Problems

This thesis dealt essentially with the study of individual instances of cryptographic algorithms. This work follows naturally from work began by Sophie Laplante, Luís Antunes and Liliana Salvador, which culminated in a Master's Thesis and the paper [ALPS07]. This paper presents several evolutions regarding the previous thesis and forms the core of Chapter 3 of the present work.

We have begun with the proposal of a notion of unconditional security of an individual instance and analysed three cryptographic systems that are known to be unconditionally secure. For each of them, we showed that if sufficiently many instances are secure, then the whole system is almost unconditionally secure under the uniform distribution, where this "almost" is measured by a constant that indicates how many bits the system leaks. This is a generalization of the notion of unconditional security, where this constant is exactly 0.

There are at least two ways in which this result can be improved. The first, it to make this parameter relative instead of absolute. To say that a system leaks 10 bits for ciphertexts that are 100-bit long is not the same as saying that the system leaks 10 bits for ciphertexts that are 10^6 bits long. More importantly, we can easily see that for certain unconditionally secure systems there will always be some instances that are individually insecure, for example, when the ciphertext is exactly equal to the plaintext. Then, for the uniform distribution, our approach will never prove a system to be absolutely secure. However, there are many questions that still need to be answered if we do not restrict ourselves to the uniform distribution. What happens to the system as a whole under a distribution that gives 0 probability to those instances that are insecure? In general, what is the average information leaked

for distributions that are not random enough, can we get better assurances than with the probabilistic proof of security? Can we determine a minimum randomness that still allows for a negligible amount of individually insecure instances? Would the traditional probabilistic proof of security still be valid for such a system? Is it possible to give a lower bound for the average information leaked by individual instances running over all computable distributions? Still regarding Chapter 3, there is still some work to do in order to adapt the analysis of authentication systems given to authentication systems with secrecy. We believe a parallel analysis is possible, resulting in theorems that would be formally very similar to the ones given in this thesis. Also, the parallel highlighted between the impersonation and the substitution attack could be developed to provide results for a generalized deception attack of any level.

We also showed how commitment systems are composed of cipher and authentication systems. There is some more work possible in here: first and foremost, to analyse the case of non optimal commitment systems. It would be interesting to see if such commitment systems can still be decomposed in different variants of cipher and authentication systems. The level of security and optimality of the different systems can be parameterized and it would be interesting to see how these parameters would be inter-related. Another interesting question is to see if it is possible to combine other primitives to produce commitment systems, instead of the cipher and authentication systems used here. For example, using multi-receiver authentication schemes where one party authenticates simultaneously to more than one verifier.

In Chapter 5, we analysed two notions of computational entropy and tried to establish their relation with the average value of K^t. Our aim was to reach a theorem similar to Theorem 2.9.30 for the computational case. We have established one of the directions, but the other still remains as an open problem. The relation between K^t and H_c is dependent on a complexity assumption, namely the existence of one-way functions. The aim of this analysis is to have a characterization of computational systems parallel to that of unconditionally secure systems. Yao had this idea in [Yao82] when he proposed to characterize security of a system with his notion of effective entropy, but he gave no proof. It remains as an open question for the future to give a detailed proof that security of a computational system can be written in terms effective entropy. After this step, if the computational parallel of Theorem 2.9.30 held, we should be able to characterize such systems at the instance level based on time-bounded Kolmogorov complexity.

This problem is still unsolved, but another approach is possible. If there is a way to show that for weak keys of a public-key ciphersystem the expected value of the

polynomial-time information given by the ciphertext about the plaintext is high, then it might be possible to show that a system where this information is low, when measured over a uniform distribution of all plaintexts, ciphertexts and keys, is secure against all adversaries of a certain kind. This would replace a computational assumption by an assumption regarding a computable value, even though this value could take a very long time to compute.

The basic idea in the analysis of instance security is to relate the security of the system to the expected value of the information in the ciphertext about the plaintext. This notion of mutual information does not easily extend to the polynomial time-bounded domain because there seems to be no symmetry of information in this setting. If there were, then there could be no one-way functions. In fact, suppose that f is a one-way function. By definition, x gives all the necessary information to compute $f(x)$ in polynomial time. If symmetry existed, then $f(x)$ would have a similar amount of information about x and since $f(x)$ can be considered of the same size of x without loss of generality, then x would be computable from $f(x)$ in polynomial time. Nevertheless, we think that time-bounded information in x about y merits further study to elicit its properties. In particular, it seems intuitive that the information in y about x that we can obtain in polynomial time should be at most the information in y about x without time limits. However, the simple generalization of using time-bounded Kolmogorov complexity on both summands in the expression of information, although it is the most natural, does not seem to have this property. Consequently, we feel that further study should be devoted to this concept.

There is a relation between public-key cryptography, one-way functions and pseudo-random generators. Public-key is possible only if trapdoor one-way functions exist, and these are a subset of standard one-way functions. If these exist, then so do pseudorandom generators and reciprocally. Further work should be devoted to characterizing one-way functions and pseudo-random generators with time-bounded Kolmogorov complexity. We would be very happy if these characterizations could preserve the distinction between weak- and strong-one way functions and the following equivalence results: that if weak one-way functions exist so do strong one-way functions (the converse is obvious) and that if one-way functions exist so do pseudo-random generators and vice-versa.

Bibliography

[AB07] Sanjeev Arora and Boaz Barak. Computational complexity: A modern approach. Draft version available online at http://www.cs.princeton.edu/theory/complexity/, 2007. last checked on 08 Oct 2007.

[AH97] Ian Anderson and Iiro Honkala. A short course in combinatorial designs, 1997. last checked on 08 Oct 2007.

[ALPS07] Luís Antunes, Sophie Laplante, Alexandre Pinto, and Liliana Salvador. Cryptographic security of individual instances. In *Proceedings of ICITS 2007, Second International Conference on Information Security*, pages 205–220, 2007.

[Ant02] Luís Filipe Coelho Antunes. *Useful Information.* PhD thesis, Faculdade de Ciências da Universidade do Porto, 2002.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. In *National Computer Conference Proceedings*, volume 48, pages 313–317, 1979.

[Blu82] Manuel Blum. Coin flipping by telephone - a protocol for solving impossible problems. In *COMPCON*, pages 133–137, 1982.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[BMSW02] C. Blundo, B. Masucci, D.R. Stinson, and R. Wei. Constructions and bounds for unconditionally secure non-interactive commitment schemes. *Designs, Codes and Cryptography*, 26(1-3):97–110, 2002.

[Bri84] Ernest F. Brickell. A few results in message authentication. *Congressus Numerantium*, 43:141–154, 1984.

[BSW03] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *RANDOM-APPROX*, pages 200–215, 2003.

[CDS99] Colbourn, Dinitz, and Stinson. Applications of combinatorial designs to communications, cryptography, and networking. In *Surveys in Combinatorics, 1993, Walker (Ed.), London Mathematical Society Lecture Note Series 187, Cambridge University Press.* 1999.

[CGH+96] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the lambert w function. *Advances in Computational Mathematics*, 5:329–359, 1996.

[Cha66] Gregory J. Chaitin. On the length of programs for computing finite binary sequences. *J. ACM*, 13(4):547–569, 1966.

[Cha75] Gregory J. Chaitin. A theory of program size formally identical to information theory. *J. ACM*, 22(3):329–340, 1975.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* John Wiley & Sons, 1991.

[Cut80] Nigel Cutland. *Computability, an introduction to recursive function theory.* Cambridge University Press, 1980.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Trans. Info. Theory*, IT-22(6):644–654, Nov 1976.

[dS90] Marijke de Soete. Bounds and constructions for authentication-secrecy codes with splitting. In *Advances in Cryptology - CRYPTO '88, Proceedings*, volume 403, pages 311–317. Springer-Verlag, 1990. Lecture Notes in Computer Science.

[GB01] Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. available online at http://www.cs.ucsd.edu/users/mihir/papers/gb.pdf, aug 2001. last checked on 08 Oct 2007.

[Gác88] P. Gács. Lecture notes on descriptional complexity and randomness, 1988. last checked on 08 Oct 2007.

[Gol01] Oded Goldreich. *Foundations of Cryptography, Volume I: Basic Tools.* Cambridge University Press, 1st edition, 2001.

[GS85] Andrew Goldberg and Michael Sipser. Compression and ranking. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 440–448. ACM Press, 1985.

[GV04] Peter Grunwald and Paul Vitanyi. Shannon information and kolmogorov complexity, 2004.

[HILL99] Johan Håstad, Russel Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HLR07] C.-Y. Hsiao, C.-J. Lu, and L. Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In *Eurocrypt 2007*, pages 169–186, 2007.

[Kah96] David Kahn. *The Codebreakers.* Scribner, 1996.

[Kat02] Jonathan Katz. Lecture notes for cmsc456, introduction to cryptography, university of maryland. available online at http://www.cs.umd.edu/ jkatz/TEACHING/crypto_F02/lectures.html, 2002. last checked on 08 Oct 2007.

[Kle52] S.C. Kleene. *Introduction in metamathematics.* Van Nostrand, Amsterdam, 2nd edition, 1952.

[Kol65] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

[Lap97] Sophie Laplante. *Kolmogorov Techniques in Computational Complexity Theory.* PhD thesis, The University of Chicago, dec 1997.

[LN83] Rudolf Lidl and Harald Niederreiter. *FINITE FIELDS.* Cambridge University Press, 1983.

[LR05] Troy Lee and Andrei E. Romashchenko. Resource bounded symmetry of information revisited. *Theor. Comput. Sci.*, 345(2-3):386–405, 2005.

[LV97] Ming Li and Paul M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications.* Springer-Verlag, New York, 2nd edition, 1997.

[Mas86] J. L. Massey. Cryptography - a selective survey. *Digital Communications*, pages 3–21, 1986.

[Mau00] Ueli Maurer. Authentication theory and hypothesis testing. *IEEE Transactions on Information Theory*, 46(4):1350–1356, 2000.

[MV04] D. McGrew and J. Viega. The galois/counter mode of operation (gcm), 2004.

[NMQO+03] Anderson C. A. Nascimento, Jörn Müller-Quade, Akira Otsuka, Goichiro Hanaoka, and Hideki Imai. Unconditionally secure homomorphic pre-distributed bit commitment and secure two-party computations. In *ISC*, pages 151–164, 2003.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *J. Comput. Syst. Sci*, 49(2):149–167, 1994.

[Pin07] Alexandre Pinto. Comparing notions of computational entropy. In *Third Conference on Computability in Europe, CiE 2007, Siena, Italy, June 2007, Proceedings*, volume LNCS 4497 of *Lecture Notes in Computer Science*, pages 606–620. Springer, 2007.

[PSMA07] Alexandre Pinto, André Souto, Armando Matos, and Luís Antunes. Commitment and authentication systems. In *Proceedings of ICITS 2007, Second International Conference on Information Security*, pages 3–23, 2007.

[Riv99] Ronald L. Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer, 1999. last checked on 08 Oct 2007.

[Sal05] Liliana Salvador. Segurança absoluta em sistemas de cifra de chave simétrica. Master's thesis, Faculdade de Ciências da Universidade do Porto, 2005.

[Sch96] Bruce Schneier. *Applied Cryptography*. Wiley Computer Publishing, 2nd edition, 1996.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, Jul and Oct 1948.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sha02] Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77:67–95, 2002.

[Sim84] G. J. Simmons. Message authentication: a game on hypergraphs. *Congressus Numerantium*, 45:161–192, 1984.

[Sim85] G. J. Simmons. Authentication theory/coding theory. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196, pages 313–317. Springer-Verlag, 1985. Lecture Notes in Computer Science.

[Sim88] G. J. Simmons. A natural taxonomy for digital information authentication schemes. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293, pages 269–288. Springer-Verlag, 1988. Lecture Notes in Computer Science.

[Sol64] R. J. Solomonoff. A formal theory of inductive inference, part i. *Information and Control*, 7(1):1–22, 1964.

[SS63] J.C. Shepherdson and H.E. Sturgis. Computability of recursive functions. *J. ACM*, 10:217–255, 1963.

[Sti87] Douglas R. Stinson. Some constructions and bounds for authentication codes. In *Advances in Cryptology - CRYPTO '86, Proceedings*, volume 263, pages 418–425. Springer-Verlag, 1987. Lecture Notes in Computer Science.

[Sti88] Douglas R. Stinson. A construction for authentication / secrecy codes from certain combinatorial designs. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293, pages 355–366. Springer-Verlag, 1988. Lecture Notes in Computer Science.

[Sti92] Douglas R. Stinson. Combinatorial characterization of authentication codes. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576, pages 62–72. Springer-Verlag, 1992. Lecture Notes in Computer Science.

[Sti02] Douglas R. Stinson. *Cryptography: Theory and Practice.* Chapman & Hall / CRC, 2nd edition, 2002.

[Tre00] Luca Trevisan. Introduction to modern cryptography. available online at http://www.cs.berkeley.edu/ luca/notes/cryptonotes99.pdf, jan 2000. last checked on 08 Oct 2007.

[Tur63] A.M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. Lond. Math. Soc.*, 42, 43:230–265 (42), 544–546 (43), 1963.

[TVZ05] Luca Trevisan, Salil Vadhan, and David Zuckerman. Compression of samplable sources. In *CCC 2004*, volume 14, pages 186–227, 2005.

[Wee04] Hoeteck Wee. On pseudoentropy versus compressibility. In *IEEE Conference On Computational Complexity*, pages 29–41, 2004.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91, 1982.