

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

325

Gilles Brassard

Modern Cryptology

A Tutorial



Springer-Verlag

New York Berlin Heidelberg London Paris Tokyo

Editorial Board

D. Barstow W. Brauer P. Brinch Hansen D. Gries D. Luckham
C. Moler A. Pnueli G. Seegmüller J. Stoer N. Wirth

Author

Gilles Brassard
Département d'Informatique et de Recherche Opérationnelle
Université de Montréal, C.P. 6128, Succursale "A"
Montréal Québec, Canada H3C 3J7

CR Subject Classification (1988): E.3

ISBN 0-387-96842-3 Springer-Verlag New York Berlin Heidelberg

ISBN 3-540-96842-3 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in other ways, and storage in data banks. Duplication of this publication or parts thereof is only permitted under the provisions of the German Copyright Law of September 9, 1965, in its version of June 24, 1985, and a copyright fee must always be paid. Violations fall under the prosecution act of the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1988

Printed in Germany

Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
2145/3140-543210

PREFACE

Dédié à Manuel Blum,

Ami inestimable

The present work grew out of lecture notes I prepared for a 3½ hour tutorial that I was invited to give at the *29th IEEE Computer Conference (CompCon)* in San Francisco on February 27, 1987. I have just completed a substantial update of the material, including the addition of several topics. My main purpose is to provide a self-contained overview of recent cryptologic achievements in a form that can be understood by readers having no previous acquaintance with cryptology. It can thus be used as preliminary reading material for an introductory class. Nevertheless, it covers enough state-of-the-art material to be also of interest to the specialist. An extensive bibliography is included.

I was originally invited to give this *CompCon* tutorial by Russel Brand. I was subsequently encouraged by David Kahn and Ronald L. Rivest to “do something” with the notes I had produced. The idea of transforming them into a monograph for Springer-Verlag’s *Lecture Notes in Computer Science* series was suggested by Lynn Montz.

Many people have generously given countless hours helping me with my undertaking. Bonnie Berger, David Chaum, Yvo Desmedt, Bennett Fox, Silvio Micali and Jean-Jacques Quisquater have provided significant feedback through several revisions. It is also a great pleasure to acknowledge the help of my co-authors for the following parts of this work: Claude Crépeau and Claude Goutier for section 5.3, David Chaum and Claude Crépeau for sections 5.5 and 5.6, and Charles H. Bennett for the whole of chapter 6. In addition, section 5.7 was written entirely by David Chaum.

David Chaum and Jean-Jacques Quisquater have made it possible for me to work on this project while I visited them at the Centrum voor Wiskunde en Informatica (CWI), Amsterdam, and Philips Research Laboratory, Bruxelles,

respectively. I am also particularly indebted to Jurjen N.E. Bos for his unstinting and never failing help at the CWI. The final manuscript was produced on the CWI Harris typesetter using their Boring computer running troff and the UNIX operating system. My research activities are supported by Canada's NSERC grant A4107.

Although they are too numerous to list explicitly, I would like to thank all those with whom I had insightful and stimulating discussions on cryptologic matters over the years. They make it an active and exciting field to work in. Last but not least, I offer all my gratitude to my wife Isabelle. Not only did she put up with my working on revising and typesetting the *CompCon* notes *immediately after* I had just finished going through the agony of producing the English version of my textbook on *Algorithmics* [52], but she actually typed in most of the original *CompCon* notes on and around New Year's Day, 1987.

It is possible that I will one day pluck up the courage to transform this monograph into a full-length textbook. In order to help me avoid carrying over errors from the former to the latter, I would be grateful to anyone kind enough to point out to me mistake(s) he/she may find in the present work, including trivial typographical errors. I would also appreciate information about the existence of final papers for the references listed here as appearing in various proceedings. Please direct all mail to me at the following address: *Département d'informatique et de recherche opérationnelle, Université de Montréal, C. P. 6128, Succursale "A", Montréal (Québec), CANADA H3C 3J7*. Thank you all in advance.

Gilles Brassard
Bruxelles, April 1988

TABLE OF CONTENTS

CHAPTER 1 — INTRODUCTION	1
CHAPTER 2 — DEFINITIONS AND CLASSIFICATION	4
CHAPTER 3 — SECRET-KEY SYSTEMS	8
3.1. Definitions and levels of attack	8
3.2. Information theory, one-time pad, and unicity distance	9
3.3. Confusion and diffusion	13
3.4. The Data Encryption Standard (DES)	14
3.5. Modes of operation	17
CHAPTER 4 — PUBLIC-KEY SYSTEMS	20
4.1. One-way functions	20
4.2. Public-key distribution	23
4.3. Public-key cryptosystems: Theory	26
4.4. The RSA cryptosystem	28
4.5. Pseudo-random bit generation	32
4.6. Probabilistic encryption	35
4.7. Hybrid systems	39
CHAPTER 5 — APPLICATIONS	40
5.1. Authentication	40
5.2. Digital signature	44
5.3. User identification	47
5.4. Coin flipping	51
5.5. Bit commitment schemes	53
5.6. Minimum disclosure proofs	57
5.7. Protection of privacy	70
5.8. Additional applications	74

CHAPTER 6 — QUANTUM CRYPTOGRAPHY 79

 6.1. Introduction 79

 6.2. Essential properties of polarized photons 81

 6.3. Quantum public key distribution 83

 6.4. Practical considerations 88

BIBLIOGRAPHY 91

CHAPTER 1

INTRODUCTION

“A man is crazy who writes a secret in any other way than one which will conceal it from the vulgar.” [9]

— Roger Bacon, *circa* 1250

For thousands of years, *cryptography* has been the art of providing secure communication over insecure channels, and *cryptanalysis* has been the dual art of breaking into such communications. Historically, *cryptology* (the combined art of cryptography and cryptanalysis) has been almost exclusively in the hands of the military and diplomats. With the advent of the computer revolution, and more importantly of a society in which vast amounts of personal, financial, commercial and technological information are stored in computer data banks and transferred over computer networks, the necessity for civilian cryptography has become overwhelming. To put it in the words of Kahn, “Cryptography, in 1945 a nation’s most closely held secret, has gone public” [155]. (See also [160, 161, 162, 185].)

Who is going to win the age-old battle between cryptography and cryptanalysis? Great (non-specialist) minds of past centuries disagree. In his *Dictionnaire philosophique* (1769), Voltaire wrote: “Ceux qui se vantent de lire les lettres chiffrées sont de plus grands charlatans que ceux qui se vanteraient d’entendre une langue qu’ils n’ont point apprise” [235] (loosely translated, this says: “Cryptanalysts are a bunch of charlatans, even more so than Champollion!”). The opposite opinion is voiced by Edgar Allan Poe in his famous tale *The Gold-Bug* (1843): “It may well be doubted whether human ingenuity can construct an enigma of this kind [a cryptogram] which human ingenuity may not, by proper application, resolve.” [190].

It is now clear that Voltaire was wrong: most historical cryptosystems have been badly broken, sometimes with spectacular consequences [154]. On the other hand, there are cryptosystems that have been proved to be unbreakable, regardless of the cryptanalyst's "ingenuity" or computing power (such as the *one-time pad*, discussed in section 3.2). Nevertheless, the question remains wide open for the more practical *public-key* cryptosystems (which is the topic of chapter 4). The current belief is that the increase in computing power witnessed in the later parts of this century places cryptographers in an unprecedentedly favourable position, to the detriment of cryptanalysts. This is ironic because the Colossus, which was the very first electronic computer in history, was built for the specific purpose of cryptanalysing German ciphers [138, 200]. (Brian Randell is reported to have once said: "By my reckoning, ENIAC was not the first computer, it was the eleventh." [250].) Thus, it may be said that cryptanalysis, being the "midwife of computer science" (Ronald L. Rivest [204]), has perhaps bred the instrument of its own doom!

Until recently, the presumed reliability of cryptosystems was "attested" by the amount of effort spent by qualified cryptanalysts in their unsuccessful attempts at breaking them. History has clearly indicated the pitfalls of this approach, as messages enciphered by cryptosystems that were believed to be invulnerable by their users were routinely decrypted. The breaking of Enigma by the Allies during (and even before) World War II is a prime example of this situation [122, 201]. The reader interested in the historical significance of cryptology is encouraged to read Kahn's wonderful account [154] and other popular books such as [246, 64, 122, 156, 92, 242].

In this century, mathematicians have worked at finding objective criteria for the security of cryptosystems, thereby transforming this ancient art into an exact science. Shannon developed information theory [214] as a result of his previous (originally classified) work on cryptography [215]. For various cryptosystems, he was able to estimate the amount of ciphertext required for a cryptanalysis to achieve any desired level of reliability. For instance, Ib Melchior could have saved a trip to Elsinore, had he only believed in Shannon's theory, when he thought he had decrypted a secret message on Shakespeare's tombstone revealing the existence of a first Hamlet edition [154].

In the past decade, computer scientists have worked at basing the security of cryptography on the more recent theory of computational complexity instead of Shannon's information theory [100, 101, 179]. The basic difference is that Shannon's theory lives on the hope that the cryptanalyst will not have enough *information* to decipher a cryptogram, while computational complexity only expects the cryptanalyst not to have enough *time* to do so.

The purpose of the present work is to give an overview of recent cryptographic achievements and techniques, and of their present and potential applications. No particular background is expected from the reader. Although the coverage of several topics is necessarily brief, an extensive (but of course not exhaustive) list of references is provided. In addition to the historical books mentioned previously, more technical books are available, such as [116, 163, 182, 94, 88, 165, 226, 194], as well as several popular [153, 111, 120, 147, 221, 51] and technical [103, 217, 169, 7, 167, 62, 142, 204] survey articles.

DEFINITIONS AND CLASSIFICATION

The purpose of a *cryptosystem* is to *encipher* an intelligible *cleartext* (also called *plaintext*), thus producing an unintelligible *ciphertext* (also called *cryptogram*). The intended receiver must be able to *decipher* the ciphertext, thus recovering the cleartext. However, *eavesdroppers* (also called *cryptanalysts*) must be unable to *decrypt* the ciphertext. Notice the important difference between deciphering and decryption.

There are several ways in which cryptosystems can be classified. We consider the following as most fundamental:

- Restricted use cryptosystems
- General use cryptosystems
 - secret-key
 - public-key.

A cryptosystem is *restricted* if its security is based on keeping secret the nature of the enciphering and deciphering algorithms. The simplest historic such system is the so-called *Cesar cipher*, which simply consists of replacing each letter in the plaintext with the third following letter in the alphabet (with wraparound when necessary). For instance, the word “cleartext” becomes “fohduwhaw”. Restricted systems are usually designed by amateurs and are almost always child’s play for professionally experienced cryptanalysts. More importantly, such systems are of no use in the modern context of a large number of users. *Codes*, which are instances of restricted cryptosystems, are not discussed here.

A cryptosystem is *general* if its security lies not in the secrecy of the enciphering and deciphering algorithms, but rather on a relatively-short secret value known as the *key*. It should be easy for individual users to come up

with their own keys so that even the designer of the cryptosystem cannot break it without knowing which key has actually been used.

For some applications (mostly military, diplomatic and covert actions), there is no reason for the designer of a general cryptosystem to publicly disclose the nature of his algorithms. Some additional safety can be obtained by keeping this information confidential. It is however crucial not to rely on this secrecy, for one never knows when it may be compromised. For this reason, reliability analyses of such systems should *always* be carried out under the assumption that the potential enemy knows all about the system, except for the actual key being used. And if the enemy in reality does not have this knowledge, so much the better. For other types of applications, such as large scale financial ones, it is in fact better to disclose how the cryptosystem works. Otherwise, users will always suspect the possible existence of a secret method to break the system.

An obvious requirement for the security of a general cryptosystem is a very large number of possible keys, so as to discourage *exhaustive search* (trying to systematically decipher a given ciphertext using each possible key until meaningful cleartext emerges). For instance, one might naively consider Caesar's cipher as an instance (with key $k = 3$) of the "general" cryptosystem consisting of replacing each letter in the plaintext with the k th following letter in the alphabet, where k is the secret key. This generalization is worthless because it accommodates only 25 non-trivial keys, making exhaustive search easy for anyone who suspects the nature of the encipherment (at least if the enciphered message has enough redundancy to allow only one meaningful decryption).

One should be aware, however, that there is no safety in large numbers alone. For instance, another generalization of Caesar's cipher consists of choosing as key an arbitrary permutation of the 26 letters of the alphabet, such as EROX...WM, and enciphering each plaintext letter according to this permutation ($A \rightarrow E, B \rightarrow R, \dots, Z \rightarrow M$) so that BAD DAY becomes REX XEW. Considering that there are $26!$ different permutations of the 26 letters, which is more than 4×10^{26} , one might feel that exhaustive search on the key space is not feasible: it would take over ten billion years to try each

possible key at the rate of one billion keys every second! Nonetheless, this (*mono-alphabetic*) *simple substitution cipher* is rather easy to cryptanalyse, if only because of the variation in natural-language letter frequencies [154, 116, 188]. Much safer cryptosystems have been designed with a significantly smaller key space.

Coming back to the classification, a general cryptosystem is *secret-key* if some secret piece of information (the key) has to be agreed upon ahead of time between any two parties that wish to communicate through the cryptosystem. In our previous example, if A enciphers a message using key EROX...WM and sends the ciphertext to B, it had better be the case that B knows which key was used for the encipherment.

This need for secure *key distribution* was not an insuperable problem in the days when cryptography was for the few, although foresight was necessary to prevent prohibitive delays before secure communication could be established. Now that cryptography has gone public, however, it is unreasonable to set up a network in which each *pair* of potential users shares a secret key in advance, because the number of keys would grow quadratically with the number of users.

In 1976, Diffie and Hellman laid the ground for overcoming this difficulty by proposing the notion of *public-key cryptography* [100, 101]. A similar idea was independently discovered by Merkle [179]. This was soon to be followed by Rivest, Shamir and Adleman's first proposed practical implementation [205]. Secure communication over insecure channels between two *totally unacquainted* parties was at last possible.

The key observation that lead to public-key cryptography was that whoever enciphers a message does not need to be able to decipher it. In such systems, each user selects a *private key* from which she obtains a pair of algorithms. She makes one of them available to everyone as her *public enciphering algorithm*, whereas she keeps secret the other one, which is the corresponding deciphering algorithm. This allows even a complete stranger to use her public algorithm to encipher a message for her; yet only she can decipher it through the use of her private algorithm. It goes without saying that such systems can

only be secure if it is infeasible to figure out a deciphering algorithm from the corresponding public enciphering algorithm.

More recently, Goldwasser and Micali have set forward the notion of *probabilistic encryption*, which is a very interesting variation on the theme of public-key cryptography [132, 133, 40]. When a message is enciphered with probabilistic encryption, it becomes essentially just as hard for a cryptanalyst to figure out *any* information on the message than it is for him to recover its *entire* contents. Moreover, there exists a probabilistic encryption scheme that is faster than the leading public-key encryption scheme proposed thus far (RSA) — see sections 4.4 and 4.6. These cryptosystems are called “probabilistic” because enciphering the same cleartext message several times under the same key can give rise to completely different ciphertexts.

Other different approaches to the key distribution problem have been proposed. For instance, Alpern and Schneider’s *keyless cryptography* can be used effectively in a network that hides the origin (but not the contents) of messages [6, 249, 93]. Shamir’s *identity based cryptosystem* removes the need for key distribution, but requires a trusted center to create private keys [211]. We shall not discuss these concepts here. Finally, Bennett and Brassard built on the work of Wiesner [237] to develop *quantum cryptography*, which proposes completely different foundations for cryptography and bases its claims of security on quantum physics rather than mathematics and computational complexity theory [23, 17, 18, 19, 20, 21, 22]. The final chapter of this book is devoted to quantum cryptography.

SECRET-KEY SYSTEMS

3.1. Definitions and Levels of Attack

A *secret-key cryptosystem* consists of a *key space* K and, for each $k \in K$, of a *cleartext message space* M_k , a *ciphertext message space* C_k , and a pair of functions $E_k : M_k \rightarrow C_k$ and $D_k : C_k \rightarrow M_k$ such that $D_k(E_k(m)) = m$ for each plaintext message $m \in M_k$. The cryptosystem is *endomorphlic* if $C_k = M_k$ for each k . Given any key k , it must be easy to obtain efficient algorithms for computing E_k and D_k . The cryptosystem is used as follows for the purpose of secure communications. If A and B expect that they might eventually have to communicate privately, they must initially agree on some secret key $k \in K$. Whenever A wishes to send a specific $m \in M_k$ to B, she uses the enciphering algorithm E_k to produce $c = E_k(m)$; she sends c to B over an insecure channel; and B uses algorithm D_k to recover $m = D_k(c)$. In many practical cryptosystems, both M_k and C_k are finite sets, often independent of the key k (such as the set of all eight-character strings). In this case, it could be that the actual message m is too long to be enciphered directly. If this occurs, m must be broken in pieces and E_k must be used several times. We discuss this situation in section 3.5.

The least one could ask of a secret-key cryptosystem is that it should be infeasible for a cryptanalyst to infer m (or, worse, k) from eavesdropping on $c = E_k(m)$. Even a cryptosystem immune to this threat may however be weak under other circumstances. Secret-key cryptography distinguishes three levels of cryptanalytic attack.

- *Ciphertext only attack*: the cryptanalyst is given $c_1 = E_k(m_1)$, $c_2 = E_k(m_2)$, $\dots$, $c_i = E_k(m_i)$, the encipherings of i distinct unknown cleartext messages under the same unknown key. He is to infer the key k or, lacking this ability, as many among $m_1, m_2, \dots, m_i$ as possible.

- *Known plaintext attack*: the cryptanalyst is given $c_1, c_2, \dots, c_i$ as above, but also the corresponding $m_1, m_2, \dots, m_i$. He is to infer k or, lacking this ability, he is to infer m_{i+1} from some *new* ciphertext $c_{i+1} = E_k(m_{i+1})$ enciphered using the same key.
- *Chosen plaintext attack*: the cryptanalyst gets to choose plaintext messages $m_1, m_2, \dots, m_i$ and he is given the corresponding $c_1 = E_k(m_1), c_2 = E_k(m_2), \dots, c_i = E_k(m_i)$. He is to infer k or, lacking this ability, he is to infer m_{i+1} from some *new* ciphertext $c_{i+1} = E_k(m_{i+1})$ enciphered using the same key. (There are real life situations in which such a powerful chosen plaintext attack can be mounted — such as “identification-friend-or-foe” systems [154].)

The difference in power between these three levels of attack is best explained through our previous example of the simple substitution cipher. When we said that it is easy to cryptanalyse, we had in mind: under a ciphertext only attack. Although this is true, it does require some work. It becomes utterly trivial to break, however, under a known plaintext attack as soon as the available cleartext messages have used at least once each letter in the alphabet (of course, all but one suffices). Patience is not even needed under a chosen plaintext attack: the key (that is, the secret alphabet permutation) yields immediately if the value of $E_k(\text{ABCD} \dots \text{WXY})$ is available.

3.2. Information Theory, One-Time Pad, and Unicity Distance

What do we mean by “it should be infeasible for a cryptanalyst to infer m ”? Two words deserve further explanation: “infeasible” and “infer”. This section is relevant mostly for ciphertext only attacks.

In the setting of Shannon’s classic information theory, “infeasible” means “mathematically impossible, regardless of available resources”. For instance, suppose you toss a fair coin and, before looking at the outcome, you ask a friend to randomly decide whether to leave it as is or to flip it over. From looking at the end result of this experiment, it is impossible to infer the original outcome of the coin toss. (We shall give quite a different meaning to the word “infeasible” when we discuss public-key systems in chapter 4.)

The exact meaning of “infer” is more difficult to make precise without introducing a substantial amount of information theory. For a formal mathematical treatment, consult [214, 215, 119]. The cryptanalyst’s ultimate goal is of course to figure out exactly and with certainty the key k or at least the plaintext message m . He may be satisfied, however, to learn some *probabilistic* information about m . Assuming the plaintext message is in English, the cryptanalyst has *a priori* information about it even *before* looking at the ciphertext. For instance, he knows that “hello” is a *more probable* heading than “xykph”. The purpose of cryptanalysis is to increase this *a priori* information by modifying the probabilities associated with each possible plaintext message, thus making the correct plaintext more probable, although not necessarily certain.

Consider a situation in which the cryptanalyst has intercepted ciphertext “xtjja” and he knows (or suspects) it was enciphered using a simple substitution cipher. This tells him that the plaintext message has five letters, the third and fourth of which being the same and the others being distinct. He cannot conclude that the plaintext is “hello” because it could also be “teddy”, for instance. Nonetheless, the *a posteriori* probabilities for these plaintexts increase relative to their *a priori* probabilities. He also knows with certainty (assuming he is correct about the nature of the cryptosystem) that the plaintext cannot be “peace” or “rambo”, and the *a posteriori* probability for these plaintexts drops down to zero, regardless of their *a priori* probabilities.

Shannon defines a cryptosystem to achieve *perfect secrecy* if knowledge of the ciphertext yields no information whatsoever on the corresponding plaintext, with the possible exception of its length. In other words, the *a posteriori* probabilities after seeing the ciphertext are *exactly* the same as were the *a priori* probabilities. Such systems do indeed exist [234] and one might wonder why they are not the ultimate solution to all cryptographic needs. There are three main reasons why this is not so. As any secret-key system, they pose the problem of key-distribution. This difficulty is amplified by a theorem of Shannon’s to the effect that perfect secrecy is *only* possible if the key space is at least as large as the cleartext message space, which amounts to saying that the secret key must be at least as long as the message itself *and* that the same

key cannot be used more than once. The third shortcoming of perfect secrecy cryptosystems is that they may be of little value for authentication purposes (as described in section 5.1).

Nonetheless, perfect secrecy cryptosystems are used in practice for very sensitive applications such as the Red Telephone between Washington and Moscow. Perfect secrecy can be achieved as follows. Let m be some cleartext message, such as “hello”. A *one-time pad* k is a purely random string of letters of equal length, such as “iwpbu”. The enciphering of message m using key k is very similar to that of our first generalization of Caesar’s cipher given in chapter 2, except that the amount by which each plaintext letter is shifted in the alphabet is not constant. In our example, plaintext letter “h” is replaced by “q”, the ninth following letter in the alphabet, because the corresponding key letter is “i”, the ninth letter in the alphabet. Similarly, “e” is shifted 23 positions with wraparound (because of key letter “w”), yielding “b”. Continuing this way, the resulting ciphertext is “qbbnj”. Notice that the third and fourth letters of the plaintext are identical, which is not the case for the corresponding letters of the ciphertext, whereas the opposite is true of the second and third letters.

It is easy to recover the plaintext “hello” from the ciphertext “qbbnj” provided the key “iwpbu” used for the encipherment is known. Without this information, however, *any* five letter plaintext could be accounted for by an appropriate key. For instance, one would decipher “qbbnj” into “peace” using key “awake”. It is crucial to understand that if the key was indeed chosen at random as it should, keys “iwpbu” and “awake” are exactly equiprobable despite the fortuitous English sound to one of them. This illustrates the importance of choosing the key at random. In fact, if both the key and the plaintext are known to be in English, a few scores of characters are usually enough to efficiently recover them both from the ciphertext [116]. To avoid correlations among ciphertexts, it is crucial that no portions of the key be re-used to encipher several messages.

The “addition modulo 26” scheme described above is fine with pencil and paper (and with the help of a so-called *Vigenère tableau* [94]). It is however more convenient to implement *electronic* one-time pads in binary: the

cleartext is transformed into a binary string through some standard encoding (which does *not* have to be secret — ASCII for instance), the one-time pad is a random binary string of the same length, and the ciphertext is the bitwise exclusive-or of these two strings (see section 3.5 for a definition of the “exclusive or”). Decipherment is obtained by the exact same process: the bitwise exclusive-or of the ciphertext and the (same) pad yields back the plaintext.

If a cryptosystem does not offer perfect secrecy, knowledge of the ciphertext yields some information on the corresponding cleartext message. With most classic secret-key cryptosystems, it is the case that the natural redundancy of English (or any natural language) makes it easier to reduce the number of candidates for the key as the length of the message increases. Consider a cryptosystem with fixed key space (the key length does not depend on the length of the cleartext message). Let $H(K)$ denote the *key space entropy* (roughly the logarithm base 2 of the number of keys) and let D measure the redundancy of the plaintext source language in bits per letter (about 3.5 for English [216]). The expected number of spurious key decipherments on length n messages is at least $2^{H(K)-nD} - 1$ for any uniquely encipherable, uniquely decipherable, endomorphic cryptosystem [12] and it is close to this value for so-called random ciphers [145].

The *unicity distance* of a cryptosystem is defined to be the length of ciphertext that is required in order to expect that there exists only one meaningful decryption for it [215]. The unicity distance does *not* tell us how much ciphertext is needed to ensure *easy* cryptanalysis, but rather how much of it is needed to have confidence in a proposed solution. For classic cryptosystems, the unicity distance is approximated by the formula $H(K)/D$ [215, 145]. For instance, the unicity distance for the simple substitution cipher is about $\log_2 26! / 3.5 \approx 25$ letters. This theoretical result is in accordance with practice as it turns out that there is nearly always a unique solution to a 30 letter cryptogram of this type, whereas it is usually easy to find a number of reasonable solutions with only 20 letters of ciphertext. Cipher A. Deavours gives a nice overview of the unicity distance calculation for several classic cryptosystems [91].

Shannon defines the term *ideal secrecy* for cryptosystems whose unicity distance is infinite, even though they may not offer perfect secrecy. With such cryptosystems, some uncertainty normally remains after a ciphertext-only cryptanalysis, however long the ciphertext is, but cryptanalysis may nonetheless bring out some information on the corresponding plaintext.

3.3. Confusion and Diffusion

The main cryptanalytic threat on conventional secret-key cryptosystems comes from the high redundancy of the source language. This allows various kinds of statistical attacks, many of which are described in Friedman's classic text [116]. Shannon suggests two basic cryptographic methods “for frustrating a statistical analysis”: diffusion and confusion [215].

The purpose of *diffusion* is to dissipate the source language redundancy found in the cleartext by spreading it out over the ciphertext. This can be achieved in two distinct ways. A *transposition* cipher rearranges the order in which letters (or bits) are written in the message. For instance, the permutation $(1 \rightarrow 3, 2 \rightarrow 5, 3 \rightarrow 4, 4 \rightarrow 1, 5 \rightarrow 2)$ applied to cleartext “hello” yields “lolhe”. Here, the secret key is the permutation and longer cleartexts are enciphered using one of the “modes of operation” of section 3.5. Although this does not affect single letter frequencies (making it rather easy to break [116]), it does obscure the frequencies of digrams, trigrams, and so on.

Another approach to diffusion is to have each letter (or bit) of the ciphertext depend on as many letters (bits) of the cleartext as possible. Consider for instance a cleartext message $m = m_1 m_2 \cdots m_n$ where each “letter” m_i is an integer between 00 and 25. Let $k = k_1 k_2 \cdots k_s$ be a secret key of the same nature for some integer s . For $0 \leq i < s$, define $m_{-i} = k_{s-i}$. Now define $c_i = (\sum_{j=0}^s m_{i-j}) \bmod 26$ for each $i \leq n$ and consider the ciphertext $c = c_1 c_2 \cdots c_n$ (see section 4.1 for the definition of the **mod** operator). It is easy to decipher when the key is known. Notice that each ciphertext letter (but the first s) depends on $s+1$ letters in the cleartext, whence diffusion. Some of the modes of operation, which we shall discuss in section 3.5, are excellent at creating diffusion.

The purpose of *confusion* is to make the relation between the key and the ciphertext as complex as possible. As a result, the cryptanalyst should not gain much useful information on the key from statistical studies of the ciphertext. This is usually brought about by the technique of substitution. The simple substitution cipher does not offer very good confusion, in particular because the most frequent letter in the ciphertext is almost certainly the fifth entry in the key (i.e.: cleartext letter “E”). It is better to use substitution on blocks of several letters, but this causes the key to be much longer (at least if the substitution is given as a table). An alternative approach is to use a different substitution for each position in the cleartext. This leads to the perfect secrecy one-time pad as well as to the ill-fated Enigma [122, 201].

Taken in isolation, neither diffusion nor confusion is a very good technique (unless the key is very long — after all, the one-time pad uses only confusion). It turns out, however, that they become much stronger when used together. The best example of this phenomenon is perhaps the Data Encryption Standard.

3.4. The Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a famous secret-key cryptosystem set forward in 1977 by the National Bureau of Standards [183]. It was designed to be used for a period of ten to fifteen years “within the Federal Government for the cryptographic protection of sensitive, but unclassified, computer data”. Although recently decertified, it is still widely used and worthy of study. Its main advantage is that it allows very high encryption and decryption speed.

We shall not describe here the details of the DES algorithm. For this, consult [183, 94]. Let us simply say that it enciphers 64-bit blocks of data using a 56-bit secret key (actually 64 bits including 8 parity bits). The DES algorithm transforms the key into sixteen 48-bit *partial keys* through a *key scheduling* algorithm that uses each of the key bits several times. After a standard *initial permutation*, the cleartext 64-bit block of data goes through sixteen *rounds* followed by the inverse initial permutation. Following Shannon’s advice, each round performs one step of confusion (using the corresponding

partial key and the so-called *S-boxes*) followed by one step of diffusion. It is remarkable that the diffusion step does not depend on the secret key; strength is gained by combining confusion and diffusion even if one of the two transformations is standard and publicly known.

The DES algorithm is designed in a way that deciphering is performed by the exact same process (in the *forwards* direction), except that the order of the partial keys is reversed in the key scheduling. This is very convenient as the same device can be used for both the enciphering and deciphering processes.

Is it safe to use the DES? In 1979, Hellman wrote a paper by the title of “DES will be totally insecure within ten years” [146]. There is a controversy about the safety of DES, which comes mostly from having a key space small enough to make exhaustive search feasible, even though rather expensive. One million processors working in parallel, each trying one million keys per second, would exhaust the key space within twenty hours. Although much studied [148, 102, 146, 230, 89, 96, 115, 72, 212, 157, etc.], no one has yet been able to find significant other loopholes in the design of DES. It thus appears to be quite adequate for privacy considerations in small to medium security applications. Its use for the purpose of authentication (section 5.1) is however more dubious because the consequences of a few successfully forged messages by a tamperer could be much more severe than that of a few successful decryptions by an eavesdropper.

A simple technique can be used to make exhaustive search more difficult, and DES should not be used without it: *multiple encipherment*. Instead of using one 56-bit key, use two (or preferably three) such keys. The obvious approach would be to encipher m as $DES_{k_1}(DES_{k_2}(m))$. However, this does not increase safety as much as it may appear at first. Indeed, both k_1 and k_2 can be computed with high probability after roughly 2^{56} DES encipherments and about the same number of DES decipherments, provided that at least two blocks of known corresponding plaintext/ciphertext are available. In order to do this, let m_1, m_2, c_1 and c_2 be such that $c_i = DES_{k_1}(DES_{k_2}(m_i))$ for $1 \leq i \leq 2$. For each key k , compute $DES_k(m_1)$ and put the result in a hash table (keeping track of the corresponding k). Then, for each key k again, compute $DES_k^{-1}(c_1)$ and look for it in the hash

table. If it is found, it is possible that the current k is in fact k_2 and that the k that was used in the first place to obtain this entry is k_1 . If, in addition, $c_2 = DES_{k_1}(DES_{k_2}(m_2))$, this pair $\langle k_1, k_2 \rangle$ is very likely to be correct (the probability of error is roughly 2^{-16}). Otherwise, resume the search for k 's such that $DES_k^{-1}(c_1)$ is in the hash table. The expected number of “false alarms” (wrong k such that $DES_k^{-1}(c_1)$ is in the table) is about 2^{48} . Notice that the hash function can be very crude since the output of DES is expected to be about random most of the time.

Although the attack described above is not much slower than exhaustive search on single encipherment, it does require significantly more central memory space to keep the table. An alternative solution is to write on magnetic tapes the value of $DES_{k_1}(m_1)$ for each key k_1 (keeping track of k_1), and on other tapes the value of $DES_{k_2}^{-1}(c_1)$ for each key k_2 (also keeping track of k_2). After sorting these tapes, a sequential pass allows to find easily the candidates $\langle k_1, k_2 \rangle$, which are then screened using m_2 and c_2 .

A slightly better use of two keys was proposed by Tuchman [229]: compute $c = DES_{k_1}(DES_{k_2}^{-1}(DES_{k_1}(m)))$. The use of DES inverse in the second stage of this formula is designed to offer compatibility with single encipherment by setting both k_1 and k_2 to the same value. Although this approach prevents the easy “meet in the middle” attack described previously, Merkle and Hellman have found a way to break it also in about 2^{56} steps (although they require a chosen plaintext attack) [181]. For this reason, they recommend the use of three independent keys and the formula $c = DES_{k_1}(DES_{k_2}^{-1}(DES_{k_3}(m)))$. Even though this makes exhaustive search currently infeasible, it is not clear whether it results in an almost invulnerable cipher: undiscovered (or unclassified) loopholes and trap-doors may exist in the DES design. Nevertheless, Coppersmith wrote the following about DES: “I am proud of my small part in this project [the design of DES]. (...) To my knowledge nobody as ever come up with a shortcut that would make cryptanalysis easier than key exhaustion.” [79].

Very high encryption and decryption speeds can be achieved with the DES when implemented in hardware. It is now possible to encipher up to 20 megabits per second [175, 11, 150, 233]. This is possible if one makes good use

of ideas such as those found in [89]. Such speeds are fast enough to encipher or decipher on the fly as data is read from or written into disk, and they are suitable for most telecommunications applications. It is possible to obtain decent speeds even by software: up to 20 kilobits per second on IBM PC and 160 kilobits per second on VAX 780 [90, 239].

3.5. Modes of Operation

Consider a cryptosystem such as the DES in which the message space consists of 64-bit blocks. How should one encipher a longer message? The obvious solution is to cut the message into 64-bit slices and encipher each of them independently using the same secret key. This idea, known as the *Electronic Code Book* (ECB) mode, is to be avoided as much as possible. Its most obvious weakness is that two identical slices of ciphertext indicate to the cryptanalyst that the two corresponding slices of plaintext are also identical. Such information can be a valuable starting point at figuring out the plaintext. The situation is even worse if DES is used for the purpose of authentication (section 5.1).

There exist at least four alternatives to the ECB mode. In all cases, two identical blocks of plaintext are (almost certainly) enciphered differently. The CBC and CFB modes (described below) make good use of the concept of diffusion: each block of ciphertext depends on all the plaintext seen so far. This makes them good for authentication purposes since it prevents the tamperer from cutting and pasting previously transmitted ciphertexts (section 5.1). Moreover, these two modes are *self-synchronizing* in the sense that only a few blocks of plaintext will be deciphered incorrectly if there is a transmission error, if the encipherment or decipherment process makes an occasional error, or even if a block of ciphertext is lost altogether without detection. The OFB and counter modes (also described below) can equally well recover from occasional transmission errors, but not so easily from the other types of errors.

Let us now briefly describe each of these modes. For more detail, consult [184, 94]. Although we shall describe them specifically for use with the DES, it should be clear that they can be used just as well for any cryptosystem whose

underlying message space is that of blocks of a given size. In the *Cipher Block Chaining* (CBC) mode, the secret key consists of the 56-bit DES key k together with a 64-bit block c_0 (although secrecy of c_0 is not essential). The plaintext m is split into 64-bit blocks $m = m_1 m_2 \cdots m_n$. For each $i \leq n$, ciphertext block c_i is computed as $c_i = DES_k(m_i \oplus c_{i-1})$, where “ $\oplus$ ” denotes the bitwise exclusive-or ($0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$ and $1 \oplus 1 = 0$). The resulting ciphertext is $c = c_1 c_2 \cdots c_n$. Given the ciphertext and knowledge of k and c_0 , decipherment is achieved by computing $m_i = c_{i-1} \oplus DES_k^{-1}(c_i)$. This formula makes it plain why errors do not propagate when using the CBC mode: plaintext block m_i depends only on ciphertext blocks c_{i-1} and c_i . A transmission error can therefore mess up only two blocks of deciphered cleartext. This mode can (and should) be used with public-key cryptosystems as well (in particular with RSA — section 4.4) when the message to be enciphered is too long to fit in a block.

In the *Cipher Feedback* (CFB) mode, the plaintext is not necessarily decomposed into 64-bit blocks. The block size is given by a parameter t such that $1 \leq t \leq 64$. The plaintext m is thus decomposed into $m = m_1 m_2 \cdots m_n$ where each m_i is a t -bit block. A 64-bit shift register is initialized to some value s_0 , which may either be part of the secret key or be sent in the clear at the outset of the transmission (so that it can be changed each time). For each $i \leq n$, ciphertext block c_i is computed as $c_i = m_i \oplus p_i$, where p_i denotes the t most significant bits of $DES_k(s_{i-1})$. The shift register is updated by discarding its t most significant bits and appending c_i to its right: $s_i = (2^t s_{i-1} + c_i) \bmod 2^{64}$. Decipherment is very similar: p_i and s_i are computed in exactly the same way and m_i is recovered as $m_i = c_i \oplus p_i$. Notice that the least significant $64 - t$ bits of each DES encipherment are discarded.

The *Output Feedback* (OFB) mode also has a variable block size t and a shift register initialized to some value s_0 . This time, however, it is necessary to use a different s_0 each time the system is used, and it should therefore be sent in the clear at the outset of the transmission (thus only the DES key k is secret). Let m be $m_1 m_2 \cdots m_n$ as usual. For each $i \leq n$, c_i is computed exactly as with the CFB mode: $c_i = m_i \oplus p_i$ where p_i denotes the t most significant bits of $DES_k(s_{i-1})$. The difference with CFB lies in the way

the shift register is updated. The official NBS mode states that $s_i = (2^t s_{i-1} + p_i) \bmod 2^{64}$, but [87, 152] describe weaknesses in this scheme. It is best to use $t \leq 8$ for the purpose of generating p_i , but it is also preferable to use $t = 64$ for the purpose of feedback. In order to meet these two contradictory requirements, it is safer to use a variant on the “official” OFB mode: compute p_i and c_i as we have indicated (with a small value for t), but then simply update s_i as $DES_k(s_{i-1})$ — note that we no longer need a shift register. In this mode, DES is used to create a pseudo-random sequence $p_1 p_2 \cdots p_n$, which is then used as a one-time pad on the cleartext (section 4.5). Because it is independent from the cleartext, the same pad is produced whenever k and s_0 are fixed. This is why s_0 should be changed each time. Decipherment should be obvious. This mode is not self-synchronizing, but it recovers nicely if an occasional block of ciphertext is received incorrectly.

An alternative to the OFB mode was suggested by Diffie: the *counter mode*. It differs from OFB only in that the shift register is replaced by a counter and its update is simply given by $s_i = 1 + s_{i-1}$, where “+” denotes ordinary arithmetic addition. If DES were a truly good enciphering algorithm, this mode should be no weaker than OFB. Moreover, only one block would be deciphered incorrectly should there be an occasional error in the DES computation.

Despite its weakness, the ECB mode is rather unavoidable for data base applications that require read/write random access to parts of the data. If only read access needs to be random, a sequentially enciphered file can nevertheless be created in either the CBC or CFB mode. *Never* use OFB or the counter mode when enciphering a file susceptible to being modified by the addition or deletion of records — see exercise 3.1 in [94].

Notice that in CFB, OFB and in the counter modes, the underlying cryptosystem (DES in our discussion) is used only in its enciphering mode. For this reason, public-key cryptosystems (section 4.3) such as RSA (section 4.4) should make use *only* of CBC, unless the necessity for read/write random access forces the use of ECB.

PUBLIC-KEY SYSTEMS

4.1. One-Way Functions

The notion of one-way and trap-door one-way functions is at the core of public-key cryptography. Consider two arbitrary sets X and Y . A function $f: X \rightarrow Y$ is *one-way* if it is easy to compute $f(x)$ for every $x \in X$, whereas it is hard for most $y \in Y$ to figure out any $x \in X$ such that $f(x) = y$ (provided at least one such x exists). This notion should not be confused with functions that are mathematically non-invertible for lack of being one-to-one or onto (that is, because there exist several different x 's such that $f(x) = y$, or because there exist none). Our current state of knowledge does not allow us to prove that one-way functions exist at all, since their existence would settle the $\mathbf{P} = \mathbf{NP}$ question [121]. Moreover, the theory of $\mathbf{NP}$ -completeness does not seem adequate to provide even mere evidence for their existence [45, 108, 141]. Nevertheless, we have *candidates* in the sense of functions that we know how to compute efficiently whereas no efficient algorithms are known (at least publicly!) to invert them.

A simple example of a candidate one-way function is *integer multiplication*. It is easy to multiply very large integers whereas even the most powerful computer with the best available algorithm is incapable of factoring a mere two hundred digit number that is the product of two roughly equal size primes. Of course, one must understand “incapable” as meaning “incapable within a reasonable time (such as one’s lifetime or the age of universe)”.

Another important example of a candidate one-way function is *modular exponentiation* (with fixed basis and modulus). Let n and a be integers such that $1 < a < n$. Let $\mathbb{Z}_n$ stand for $\{0, 1, 2, \dots, n-1\}$. Modular exponentiation (with respect to basis a and modulus n) is the function $f_{a,n}: \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ defined by $f_{a,n}(m) = a^m \bmod n$, where “ $i \bmod j$ ” stands for the positive remainder of i when divided by j . It is not immediately obvious that this

function can be computed efficiently when all three parameters (a , n and m) are several hundred digit long. The fact that this is possible is best seen through an example: $a^{25} = (((a^2 \times a)^2)^2 \times a$. This shows how to compute a^{25} in only four squarings and two further multiplications. In a computation of $a^m \bmod n$, the reduction modulo n should be done after each squaring and each multiplication in order to avoid the accumulation of a very large integer. If the exponent m is l -bit long, the following recursive algorithm requires between l and $2l$ modular multiplications (counting the squarings as multiplications) in order to compute $a^m \bmod n$:

```

function expo( $a, m, n$ )
    if  $m = 0$  then return 1
    if  $m$  is even then return  $[expo(a, m/2, n)]^2 \bmod n$ 
    {otherwise} return  $[a \times expo(a, m-1, n)] \bmod n$  .
    
```

By analogy with real analysis, the reverse operation is known as the *discrete logarithm problem*: given integers a , n and x , find some integer m such that $a^m \bmod n = x$ (if one exists). For example, $5^4 \bmod 21 = 16$, hence 4 is a solution to the discrete logarithm of 16 in base 5, modulo 21. In this case, 10 is also a solution since $5^{10} \bmod 21 = 16$ as well. You may wish to verify that 3, on the other hand, has no logarithms base 5, modulo 21. Although large modular exponentiations can be carried out efficiently, no algorithm is currently known to exist for the computation of large discrete logarithms in a reasonable time, even on the fastest computers. Even though we cannot prove either that such algorithms do not exist, it is conjectured that modular exponentiation (with fixed basis and modulus) is indeed a one-way function.

Obviously, one-way functions cannot be used directly as cryptosystems (i.e.: enciphering m as $f(m)$) because even the *legitimate* receiver would be unable to recover the plaintext! We shall see later that they are nonetheless useful for password protection (section 5.3). A notion more relevant to cryptography is that of *trap-door one-way functions*. A function $f: X \rightarrow Y$ is trap-door one-way if it can be computed efficiently both forwards and backwards, yet one can disclose an efficient algorithm to compute f such that even complete knowledge of how this algorithm works does not enable one to figure

out an efficient algorithm to reverse the computation. The secret that enables one to obtain both efficient algorithms is known as the *trap-door*.

Our first candidate trap-door one-way function is very much like our second candidate for a one-way function: modular exponentiation *with fixed exponent and modulus*. Let m and n be integers and define $\mathbb{Z}_n$ as before. The modular exponentiation (with respect to exponent m and modulus n) is the function $g_{m,n}: \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ defined by $g_{m,n}(a) = a^m \bmod n$. Make sure you understand the difference between the functions $f_{a,n}$ and $g_{m,n}$. By analogy with real analysis again, the reverse operation is known as *taking the m th root of x modulo n* : given integers m , n and x , find some integer a such that $a^m \bmod n = x$ (if one exists). For example, 5 is a 4th root of 16 modulo 21 since we have already seen that $5^4 \bmod 21 = 16$. Obviously, 2 is also a 4th root of 16 modulo 21. Can you find the other 4th roots of 16 modulo 21? Find an integer x with no 4th roots modulo 21.

Whenever the exponent m and the modulus n are fixed, we have seen an efficient algorithm for computing $g_{m,n}(a)$ for any base a . Contrary to the discrete logarithm problem, however, we know that there also exists an efficient algorithm for taking m th roots of x modulo n (or finding out that none exist) given any x . The curious phenomenon is that it is not known how to efficiently construct this efficient algorithm given only m and n . In other words, the function $g_{m,n}$ is definitely *not* one-way since we know it can be inverted efficiently, despite the fact that we do not know how to do so! Nonetheless, it is easy to construct an efficient algorithm for taking m th roots modulo n provided the decomposition of n into prime factors is known. For this reason, $g_{m,n}$ is a candidate trap-door one-way function for which m and n are used as public information whereas the factoring of n serves as secret trap-door. We shall see how this can be used when we study the famous RSA cryptosystem (section 4.4).

An important special case of modular exponentiation occurs when the exponent is 2 and when the modulus is of a special form. More number theory is necessary to understand this second example of a candidate trap-door one-way function. If p and q are two distinct large primes of roughly equal size and if both p and q are congruent to 3 modulo 4, we say that their

product $n = pq$ is a *Blum integer*. Define $\mathbb{Z}_n^*$ as the set of integers between 1 and $n - 1$ that are divisible by neither p nor q . Finally, define QR_n as the subset of $\mathbb{Z}_n^*$ consisting of numbers that are perfect squares modulo n . Elements of QR_n are known as *quadratic residues* modulo n . As a toy example, consider $p = 19$ and $q = 23$, thus $n = 437$. Here, 135 is a member of $\mathbb{Z}_n^*$, whereas 133 is not (because $133 = 19 \times 7$). Moreover, 135 is not a quadratic residue modulo 437 because there does not exist an integer a such that $a^2 \equiv 135 \pmod{437}$, whereas 139 is a quadratic residue because $24^2 = 576 \equiv 139 \pmod{437}$.

Let us state without proofs a few relevant theorems. The number of elements in $\mathbb{Z}_n^*$ is $(p - 1)(q - 1)$ and exactly one fourth of them are quadratic residues. Each quadratic residue admits precisely four distinct “square roots” in $\mathbb{Z}_n^*$, of which exactly one is itself a quadratic residue. We call this particular square root the *principal* square root. In our example, 24 is the principal square root of 139 modulo 437 and the other three square roots are 185, 252 and 413. The cryptographically significant fact is that the ability to extract square roots modulo n is *computationally equivalent* to the ability to factor n . In other words, whoever knows the factors of n can efficiently compute principal square roots modulo n , whereas such computations are as hard as factoring n for those who do not already know the factors.

Our second candidate trap-door one-way function should now be obvious. Someone chooses p and q at random and computes $n = pq$, which he publicly discloses. From now on, anyone can efficiently compute squares modulo n , but only our friend can efficiently invert this operation (assuming factoring is hard). In this example, the public information is n and the secret trap-door is again its factorization.

For more background on computational number theory, consult [165].

4.2. Public-Key Distribution

We have seen that one of the major difficulties with large scale multi-user secret-key cryptosystems is that each pair of users must share a secret key. Assume to the contrary that two given users initially share no secret information, and that they suddenly wish to establish secure communications between

them. The conventional solution would be for them to meet physically in order to exchange a secret key, or to make use of some sort of trusted courier. Both these solutions are slow and expensive, and they may not be all that safe. The purpose of a *public-key distribution system* is to allow two such users to come up with a secret key as the result of a discussion over an insecure channel, in a way that an eventual eavesdropper cannot figure out the key even after listening to the entire discussion.

More precisely, we wish a protocol by which A and B exchange messages m_1 (from A to B), m_2 (from B to A), $\dots$, until eventually A and B agree on some key k , in a way that it is infeasible to infer k from knowledge of m_1 , m_2 , $\dots$ alone. Let us stress again that this must be achieved even though A and B share no information beforehand that is unknown to the eavesdropper.

The first protocol to achieve this seemingly impossible goal was proposed by Diffie and Hellman [101] in 1976. It is based on the discrete logarithm problem introduced in section 4.1. Let n be some large integer and let g be another integer strictly between 1 and $n - 1$. As a first step of the protocol, A and B agree on n and g over the insecure channel (alternatively, n and g could be *standard* parameters used by all users of the system). Then, A chooses some large integer x and computes $X = g^x \bmod n$. Similarly, B chooses y and computes $Y = g^y \bmod n$. At this point, A and B exchange X and Y over the insecure channel *but they keep x and y secret* (only A knows x and only B knows y). Finally, A computes $Y^x \bmod n$; similarly, B computes $X^y \bmod n$. Both these values are equal since it amounts to $g^{xy} \bmod n$ either way. This is the key k they wished to establish in common.

The eavesdropper is faced with the task of figuring out k from the information sent over the insecure channel: g , n , X and Y . The obvious approach for the eavesdropper would be to figure out x from g , n and X (or at least some $\hat{x}$ such that $g^{\hat{x}} \bmod n = X$ as any such $\hat{x}$ would yield $Y^{\hat{x}} \bmod n = k$). However, this is precisely the discrete logarithm problem, which is believed to be infeasible. No one has yet figured out a way of computing k efficiently from g , n , X and Y , but no one has either been able to prove that this is not possible or even that there is no better way to do so than to first compute a

discrete logarithm. It is hence conceivable that the computation of k could be carried out efficiently even if the discrete logarithm problem should be genuinely infeasible.

The choice of g and n can have a substantial impact on this scheme's efficiency and safety. In order to increase the range of potential final values for k , it is important that the modular exponentiation function $f_{g,n} : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ (as defined in section 4.1) be as nearly one-to-one as possible. Whenever n is a prime number, there always exist g 's such that $g^x \bmod n$ takes every value between 1 and $n - 1$ as x covers the same range. Such g 's, known as *generators for the cyclic group $\mathbb{Z}_n$* , should be sought. It is safer in this case to choose n such that $(n - 1)/2$ is also a prime [191]. Alternatively, one could perform all the calculations in the Galois field $\text{GF}(2^k)$, a technique that goes far beyond the scope of this overview [28]. This allows a much more efficient computation of the multiplication (hence of the exponentiation) because it avoids the need for computing carries and for modular reductions. The key size should however be longer in this case.

Although very large discrete logarithms may be infeasible to compute, the reader should be warned that there exist algorithms far better than exhaustive search for this task. The best currently known algorithms are described in [80] when n is a prime number and in [186] for computations in $\text{GF}(2^k)$. This should be carefully considered when choosing a key size. An ill-fated hardware implementation was once built by Hewlett-Packard using $\text{GF}(2^{127})$ [247]. This key size was readily broken by [29].

Assuming n and g are standard parameters, an interesting alternative to the interactive protocol described above is to set up and distribute once and for all a *directory*. Each user enters in this directory his own X computed as $g^x \bmod n$ from his randomly chosen secret x . This allows any two users to come up with their shared secret key before they even get to talk to one another for the very first time. A major disadvantage of this directory approach is that it does not encourage users to change their secret keys often enough. We shall come back to this point in section 4.7.

A different approach to the key distribution problem is provided by Bennett and Brassard's *quantum cryptography*. Its safety does not depend on

unproved computational complexity assumptions about the difficulty of computing certain functions. It would hold even if the eavesdropper had infinite computing power or should $\mathbf{P} = \mathbf{NP}$ [17, 18, 19, 20, 21]. The last chapter of this tutorial describes quantum public-key distribution.

4.3. Public-Key Cryptosystems: Theory

A public-key distribution system such as the one described in section 4.2 allows two parties to come up with a shared piece of secret information. However, neither party has any direct influence on what this information will turn out to be. Should A wish to send one *specific* message m to B, the use of a public-key distribution system would have to be followed by that of a secret-key cryptosystem in which the initially shared information would serve as secret key.

Public-key cryptosystems are used *directly* to encipher meaningful messages. Very much like secret-key cryptosystems, a *public-key* cryptosystem consists of a *key space* K and, for each $k \in K$, of a *cleartext message space* M_k , a *ciphertext message space* C_k and a pair of functions $E_k : M_k \rightarrow C_k$ and $D_k : C_k \rightarrow M_k$ such that $D_k(E_k(m)) = m$ for each plaintext message $m \in M_k$. Again as in secret-key cryptosystems, it must be easy to obtain efficient algorithms for computing both E_k and D_k from any key k . We shall refer to the algorithms thus obtained as the *natural algorithms*. The important new twist is that E_k must be trap-door one-way: it must be infeasible to infer *any* efficient algorithm for computing D_k (not only the natural algorithm) from knowledge of a description of the natural algorithm for computing E_k . For this to be possible at all, of course, it must be that k itself cannot feasibly be inferred from a description of the natural algorithm for computing E_k . In particular, this means that k *must not* appear in the clear in the natural enciphering algorithm.

A public-key cryptosystem is used as follows. Once and for all, each user selects a random $k \in K$. He uses it to obtain both natural algorithms for E_k and D_k . He makes his enciphering algorithm E_k publicly available, perhaps through the use of some directory, but he keeps his deciphering algorithm D_k secret. Whenever another user wishes to send him a message, she looks up in

the directory for his public enciphering algorithm and she uses it to encipher her message. Using his secret trap-door, only the legitimate receiver can decipher the message.

Contrary to secret-key cryptosystems, observe that if user A enciphers some message m for user B, keeps the ciphertext c but loses the cleartext (and forgets all about its contents), she is no better off than an eavesdropper to figure out m from c . Also contrary to secret-key cryptosystems, if an eavesdropper intercepts a ciphertext and if he knows for whom it was enciphered, he can use the public enciphering algorithm to verify any specific guesses as to what the cleartext might be. The cryptanalyst's ability to come up with ciphertexts for cleartext messages of his choice causes the collapse of the classic levels of attack discussed about secret-key cryptosystems (section 3.1). However, the following meaningful attack can sometimes be mounted:

- *Chosen ciphertext attack*: against a given user whose functions are E_k and D_k , the cryptanalyst gets to choose ciphertext messages $c_1, c_2, \dots, c_i$ and he is given the corresponding $m_1 = D_k(c_1), m_2 = D_k(c_2), \dots, m_i = D_k(c_i)$, provided they exist. He is to infer k or any efficient algorithm for computing D_k or, lacking this ability, he is to infer m_{i+1} from some *new* ciphertext $c_{i+1} = E_k(m_{i+1})$.

Public-key cryptosystems can only exist if both one-way *and* trap-door one-way functions exist. Indeed, the enciphering functions must be trap-door one-way and the process that yields the natural enciphering algorithm from the key must implement a one-way function. As a result, we do not know how to prove the existence of public-key cryptosystems. It could be nothing more than a fancy way of talking about the empty set, even though the feasibility of the concept has been formally demonstrated in relativized settings [49, 46, 47].

Just as was the case with (trap-door) one-way functions, we must therefore be content with candidates for public-key cryptosystems. The two best known such candidates were designed shortly after Diffie and Hellman set forward the notion of public-key cryptography [100, 101]. One of them, Merkle and Hellman's so-called *knapsack* cryptosystem [180, 147] was eventually broken [99, 104, 209, 3, 59, 166, 60]. Although there still are unbroken variations

on the original scheme, it does not seem advisable to trust them. Brickell has written a very detailed history of the cryptanalysis of the knapsack cryptosystem [61]. The other best known public-key cryptosystem remains yet undefeated and we shall now describe it in some detail. (Other systems have been proposed, such as those of McEliece [177] and ElGamal [105]; we shall not discuss them here. For a comprehensive overview of public-key cryptography, consult [167].)

4.4. The RSA Cryptosystem

The very first public-key cryptosystem ever proposed in the open literature was that of Rivest, Shamir and Adleman [205]. It has come to be known as the RSA or MIT cryptosystem. It is based on the belief that modular exponentiation with fixed exponent and modulus is a trap-door one-way function (section 4.1). For an early easy-to-read account of it, consult [120].

Let p and q be two large distinct primes. Let $n = pq$ and let e be some integer relatively prime to $(p-1)(q-1)$. Each such triple $k = \langle p, q, e \rangle$ is a *private key* for the RSA cryptosystem. Both the corresponding cleartext and ciphertext message spaces M_k and C_k are $\mathbb{Z}_n$, the set of non-negative integers smaller than n . Should the actual message be too long to belong to $\mathbb{Z}_n$, it would have to be broken into pieces and enciphered using the Cipher Block Chaining mode described in section 3.5. The enciphering function corresponding to key k is $E_k : M_k \rightarrow C_k$ defined by $E_k(m) = m^e \bmod n$. In order to specify the natural algorithm to compute it, it suffices to write down e and n in the public directory. This pair $\langle e, n \rangle$ is known as the *public key*, and it is easily computed from the private key $\langle p, q, e \rangle$.

Recall from section 4.1 that E_k is a candidate trap-door one-way function because, although there exists an efficient algorithm to compute its inverse D_k , we do not know how to obtain it efficiently given only the natural algorithm to compute E_k (that is, given only n and e). However, an efficient algorithm to compute D_k is easy to obtain given the additional private information p and q . For this purpose, an extension of Euclid's greatest common divisor algorithm is used to compute an integer d such that $ed \equiv 1 \pmod{\phi}$, where $\phi = (p-1)(q-1)$. By a theorem of Euler's, $m^{ed} \equiv m \pmod{n}$ for every

integer m , and hence $m^{ed} \bmod n = m$ provided $0 \leq m < n$, that is provided $m \in M_k$. The deciphering function is therefore $D_k : C_k \rightarrow M_k$ defined by $D_k(c) = c^d \bmod n$, and the efficient algorithm for modular exponentiation can also be used to compute it. (Based on the Chinese remainder theorem, an even more efficient algorithm exists to compute D_k given p and q [193].)

To summarize, each user of the RSA cryptosystem must once and for all select appropriate integers p , q and e at random, and compute d from them. He then makes his public key $\langle e, n \rangle$ available in the users' directory, whereas he keeps d secret. This enables any other user to encipher messages that only he can decipher. For this idea to be of practical use, it is crucial that the generation of large random primes and the computation of d be feasible. Fortunately, primality testing appears to be genuinely easier than factoring thanks to the probabilistic algorithms of Solovay-Strassen and of Rabin [223, 197]. For the cryptographically more relevant question of prime *generation* (rather than *testing*), consult [81, 13]. For a description of Euclid's extended algorithm for the computation of d , consult [4, 94] (alternatively, if you prefer to rediscover Euclid's extended algorithm by yourself, you may find useful the hints given in Problem 8.5.12 of [52]).

As a toy example, let $p = 19$ and $q = 23$ so that $n = 437$ and $\phi = 396$. Let $e = 13$ so that $d = 61$ since $13 \times 61 = 793 = 2\phi + 1$. The cleartext message $m = 123$ is then enciphered as $c = 123^{13} \bmod 437 = 386$ and indeed $386^{61} \bmod 437 = 123$.

Let us stress that a cryptanalyst having eavesdropped on ciphertext $c = E_k(m)$ sent to a given user knows the natural enciphering algorithm used by the sender to compute E_k . This has two important consequences. Should the eavesdropper guess exactly the cleartext message m , he could compute $E_k(m)$ as well as the sender and compare the result with c to check his guess. This threat is serious if the number of possible cleartext messages is small enough to allow exhaustive search. Padding short messages with random bits may solve this difficulty to some extent but use of *probabilistic encryption* (section 4.6) is a much better solution. See also [206].

The other consequence of the eavesdropper's knowledge of the public enciphering algorithm is more specific to the RSA cryptosystem. He knows

that $c = m^e \bmod n$ for known values of c , e and n (but unknown m). If he could factor n , (thus recovering the legitimate receiver's private key $\langle p, q, e \rangle$), he could obtain $\phi = (p-1)(q-1)$ and apply Euclid's extended algorithm to compute d and then $m = c^d \bmod n$. Fortunately, no known algorithm can factor a two hundred decimal digit number in a reasonable amount of time, and it is thus considered quite safe to choose both p and q about one hundred digits long. Care should be taken in the choice of p and q , so as to give no handle to the known factoring algorithms. In particular, the greatest common divisor of $p-1$ and $q-1$ should be small and both $p-1$ and $q-1$ should have large prime factors. Blakley and Borosh suggest one should choose p and q so that $(p-1)/2$ and $(q-1)/2$ are prime [31]. For an extensive discussion of these issues, consult [94]. Gordon suggests efficient techniques to choose suitably strong primes [139].

Even should factoring be genuinely hard, breaking RSA is not known to be equally hard. It is possible that d can be computed efficiently from the public information e and n with no need for the factors of n . It is also possible that the value of d (hence the factors of n) is genuinely infeasible to recover from e and n , yet there is another efficient algorithm to recover m from e , n and $m^e \bmod n$. Other public-key cryptosystems have been proposed for which recovering the cleartext from the eavesdropper's available information has been proved to be as hard as factoring large integers [196, 240], but they fall immediately against a chosen ciphertext attack. Finally, even if m were genuinely infeasible to compute from the information available to the eavesdropper, it might still be easy to obtain efficiently some *partial* information such as half the bits of m . Probabilistic encryption solves all these potential weaknesses under the mere assumption that factoring is hard.

Users of RSA should be aware that this cryptosystem is weak under some forms of chosen ciphertext attacks [86, 95]. Assume for instance that an eavesdropper intercepted some $c = m^e \bmod n$, where e and n are public. He would like to figure out m . Under a chosen ciphertext attack, he is allowed to give any $\hat{c}$ to the legitimate receiver so as to obtain the corresponding $\hat{m}$ such that $\hat{c} = \hat{m}^e \bmod n$. It is reasonable to expect that the receiver would refuse to cooperate if the eavesdropper tried directly with $\hat{c} = c$ (otherwise, no

cryptosystem can possibly be safe). However, the eavesdropper can *hide* his question by randomly choosing some $x \in \mathbb{Z}_n^*$ and computing $\hat{c} = x^e c \bmod n$. The original plaintext m can then be obtained efficiently (using Euclid's extended algorithm) because $m = \hat{m} x^{-1} \bmod n$. It is not known if a more subtle chosen ciphertext attack could *really* break RSA in the sense of allowing the eavesdropper to figure out the factors of n or at least the secret deciphering exponent d . If this were so, later ciphertexts could be decrypted without need for the legitimate receiver's collaboration. For a first step in this direction, consult [98].

Despite all its advantages over secret-key cryptosystems, RSA is substantially slower than the DES. Recall that hardware DES implementations can currently achieve a rate of 20 megabits per second. This makes the fastest RSA encryption devices commercially available look rather bad. Kochanski's gate array [159, 222] can achieve about 5 kilobits per second for a 512-bit key (which is the absolute lowest limit on key size one should consider). A faster implementation due to Omura is reported in [185], but it is still more than a thousand times slower than DES. There is also the famous "Rivest Chip" that was never quite debugged [202, 203] and Brickell's design for an implementation that could theoretically achieve 25 kilobits per second. An even faster implementation is reported in [207]. Read also [187]. Needless to say that software implementations are even slower. The fastest commercially available RSA encryption the author is aware of for the IBM PC is Algorithmic Research Ltd.'s CryptoCom. It enciphers very rapidly because it always uses exponent 3 as the public key (which is very dangerous normally [143], but not so much in this context because RSA is only used to exchange DES keys). It decipheres one 512-bit block in merely 9 seconds for a record breaking speed of about 57 bits per second! (In fact, CryptoCom is a hybrid system so that it goes quite a bit faster in the long run — see section 4.7). Much faster RSA implementations on IBM PC have been claimed, but official documentation about them is not yet available.

4.5. Pseudo-Random Bit Generation

A sequence is *pseudo-random* if it *appears* to be patternless and random, although it was in fact produced by a purely deterministic process known as a *pseudo-random generator*. Such generators are given a truly random starting sequence known as the *seed*, and they are to deterministically produce from it a much longer pseudo-random sequence. In this sense, one may view pseudo-random generators as *amplifiers* of randomness. For an encyclopaedia on the classic generation of pseudo-random sequences and for tests whose purpose is to tell apart pseudo-random sequences from truly random ones, consult [158].

Randomness and cryptography are very strongly related. The prime purpose of cryptosystems is to transform non-random meaningful cleartexts into an apparently random jumble. This capability can be used for the generation of pseudo-random sequences as follows. Let E_k be some encipherment algorithm and let x_0 be any cleartext. Consider the sequence defined by $x_{i+1} = E_k(x_i)$ for $i > 0$. If E_k is good for cryptographic purposes, it is likely that $x_1, x_2, \dots$ is a sequence with no apparent patterns (although it is bound to be eventually cyclic). To reduce the correlations in this sequence, it may be preferable to keep only a few bits from each x_i .

The relation between randomness and cryptography in the other direction is more interesting. Even the best cryptosystem would be worthless if the cryptanalyst could guess the key being used (this remark applies just as well for secret-key and public-key cryptosystems). There is no better way to prevent this threat than to choose the key purely at random. Failure to do so in the “telegram keys” was a major cause in the breaking of Enigma [122, 201]. As an extreme example, consider the one-time pad described in section 3.2. We have seen that it offers perfect secrecy if the pad is chosen at random, whereas it can be broken without much difficulty if the pad is in plaintext English. Recall that the main inconvenience of one-time pads is that the pad must be not only random but just as long as the message to be enciphered, and used only once.

The Output Feedback mode of section 3.5 puts these two concepts nicely together: it uses a cryptosystem to generate a pseudo-random sequence from

a short seed (k and s_0 , of which only k is secret), and this sequence is used as a one-time pad on the actual cleartext message. The advantage of this approach is that the secret key k is much shorter than the cleartext and it can be re-used several times as long as s_0 is changed each time (recall that s_0 is given in the clear as part of the ciphertext, so that only k needs to be safely exchanged once and for all). The unavoidable price to pay for using a shorter key is of course the loss of perfect secrecy. But is this Output Feedback mode anything more than a wishful thinking heuristic?

Let us say that a pseudo-random generator is *cryptographically strong* if the sequence it produces from a short secret seed is essentially as good as a truly random sequence for the purpose of being used as a one-time pad. By “essentially as good”, we mean that no *feasible* computation can allow the cryptanalyst to learn *any* information on the cleartext from eavesdropping on the ciphertext (except with a vanishingly small probability). In other words, it behaves as if it offered Shannon’s perfect secrecy as long as the cryptanalyst does not spend an inordinate amount of time. Such generators can be used to implement a secret-key cryptosystem if both parties agree on the secret seed they are going to use, provided they never use the same seed twice. It is much less obvious that cryptographically strong pseudo-random generators can be used to implement *public*-key cryptosystems, but we shall see in the next section that such can be the case.

A first step towards the establishment of such generators was taken by Shamir [210]. The key notion of pseudo-random generators *unpredictable to the left* was subsequently introduced by Blum and Micali [41]: a cryptanalyst who knows how the generator works but does not know which seed is actually used cannot do better than tossing a fair coin in order to guess the first bit produced by the generator upon seeing the sequence generated subsequently (unless he is very lucky or willing to undergo an unfeasible computation). As usual, we do not know if such generators exist but a first candidate was also proposed by Blum and Micali [41], who proved that their generator is unpredictable to the left under the assumption that it is infeasible to extract discrete logarithms. The full relevance of pseudo-random generators unpredictable to the left was established by Yao, who proved that any such

generator is cryptographically strong [243]. Finally, Levin has given necessary and sufficient conditions for the existence of such generators [170].

We now describe a simpler and computationally more efficient candidate cryptographically strong pseudo-random generator, which is known as the BBS generator, named after Blum, Blum and Shub [33]. It is based on the second candidate trap-door one-way function described in section 4.1. Recall that n is a *Blum integer* if it is the product of two distinct primes p and q , both of which are congruent to 3 modulo 4. Recall also that squaring modulo n is a permutation of the quadratic residues modulo n , and that it is believed to be a trap-door one-way function because the difficulty of inverting it (i.e.: of computing *principal square roots*) has been proved to be computationally equivalent to factoring n . A stronger theorem can be proved under the assumption that factoring n is hard: for nearly all quadratic residues x , tossing a fair coin yields the best possible estimate on the least significant bit of x that can feasibly be computed upon seeing $x^2 \bmod n$ [231, 232, 5]. In other words, not only is the principal square root infeasible to compute as a whole, but so is even probabilistic information on its least significant bit (such as: “I am not certain, but I believe this bit is more likely to be a zero than a one”).

We are now ready to describe the BBS generator and to prove under a suitable assumption that it is cryptographically strong. Let n be a Blum integer of unknown factorization. Choose as seed a random quadratic residue x_0 (in order to do this, choose a random integer x relatively prime to n and compute $x_0 = x^2 \bmod n$). For $i \geq 0$, define recursively $x_{i+1} = x_i^2 \bmod n$ and let b_i be the least significant bit of x_i . For any integer t , the first t bits generated from seed x_0 is defined to be the pseudo-random sequence $BBS_{n,t}(x_0) = b_0 b_1 b_2 \cdots b_{t-1}$.

To say that the BBS generator is unpredictable to the left means that one cannot guess b_0 upon seeing n and $b_1 b_2 b_3 \cdots b_{t-1}$. If this were not so, the least significant bit of the unknown principal square root x of any given quadratic residue y could be estimated as follows. Compute the pseudo-random sequence $BBS_{n,t-1}(y)$ and *pretend* it is in fact $BBS_{n,t}(x)$ with its first bit removed. Guess this missing bit and notice it is by definition the least

significant bit of x we were seeking. It follows that the BBS generator must be unpredictable to the left under the assumption that it is hard to factor n . Hence, by Yao's theorem, it is cryptographically strong.

An additional nice property of this generator is that it gives *direct access* to the individual bits it produces for whoever knows the factorization of n . For this, notice that $x_i = x_0^{2^i} \bmod n$. By Euler's theorem, $x^\phi \equiv 1 \pmod{n}$, where $\phi = (p-1)(q-1)$. Therefore, $x_i = x_0^{2^i \bmod \phi} \bmod n$ is computed efficiently from the seed x_0 and the desired bit index i through two applications of the fast modular exponentiation algorithm.

4.6. Probabilistic Encryption

Public-key cryptography solves to a large extent the key distribution problem, which is so severe with secret-key cryptography. However, as we pointed out above, eavesdropping on the ciphertext $c = E_k(m)$ always leaks some information about the plaintext m because the cryptanalyst can compute for himself the public enciphering function E_k on whichever plaintexts suit him. Given any $\hat{m}$ of his choice, he can easily decide whether $m = \hat{m}$ since this is so if only if $E_k(\hat{m}) = c$. Even if recovering m from c were indeed infeasible from knowledge of the natural enciphering algorithm only, which we do not know how to prove, there is no say as to how much and which partial information leaks about m . To use an image of Goldwasser's, using public-key cryptography is like hiding a camel by throwing a blanket over it: it may conceal which camel it actually is, but not its number of humps.

The purpose of *probabilistic encryption*, a notion invented by Goldwasser and Micali [132], is to encipher messages in a way that no feasible computation on the ciphertext can yield any information whatsoever on the corresponding plaintext (except with a vanishingly small probability). This is reminiscent of perfect secrecy systems in the sense of Shannon's, with the additional benefits of short keys and the possibility for each user to disclose his public enciphering algorithm. Of course, these systems cannot offer true perfect secrecy; they are totally insecure against a cryptanalyst with unlimited computing power.

The main technical difference between probabilistic encryption and public-key cryptosystems is that the natural enciphering algorithms are now probabilistic rather than deterministic: the same cleartext message can give rise to a large number of distinct ciphertexts. As a result, a cryptanalyst having a candidate for the correct plaintext could no longer verify his guess by enciphering it (with the legitimate receiver's natural algorithm) and comparing the result with the intercepted ciphertext.

Formally, a probabilistic encryption system consists of a *key space* K and, for each $k \in K$, of a *cleartext message space* M_k , a *ciphertext message space* C_k , a *randomness space* R_k and a pair of functions $E_k : M_k \times R_k \rightarrow C_k$ and $D_k : C_k \rightarrow M_k$ such that $D_k(E_k(m, r)) = m$ for every plaintext message $m \in M_k$ and randomness source $r \in R_k$. From any $k \in K$, it must be easy to obtain both efficient natural algorithms for computing E_k and D_k , but it must be hard to figure out any efficient algorithm for computing D_k given only the natural algorithm for computing E_k .

A probabilistic encryption system is used in a way very similar to public-key cryptosystems. Once and for all, each user selects a key $k \in K$, which he uses to obtain both natural algorithms E_k and D_k . He makes the enciphering algorithm E_k publicly available and keeps secret his deciphering algorithm. Whenever another user wishes to send him a message m , he looks up for E_k in the directory, he randomly chooses some $r \in R_k$, and he computes the ciphertext $c = E_k(m, r)$. Using his secret trap-door, only the legitimate receiver can feasibly recover m from c .

As a consequence of the large number of ciphertexts corresponding to each plaintext, some amount of *data expansion* is unavoidable when using probabilistic encryption: the ciphertext is always longer than the corresponding cleartext. Notice that this was not so with the RSA cryptosystem. Despite all its nice theoretical properties, the original Goldwasser-Micali probabilistic encryption scheme [133] suffered such a tremendous data expansion that it was not of much practical value. We shall not describe it here. Fortunately, probabilistic encryption has come of age for there now exists a scheme even more efficient than RSA. For the purpose of privacy (not authenticity — see section 5.1), this scheme of Blum and Goldwasser's

(described below) is the best that academia has had to offer thus far. It is based on the belief that squaring modulo a Blum integer is trap-door one-way (last example in section 4.1) and on the cryptographically strong pseudo-random bit generator described in section 4.5. Intuitively, the BBS generator is used by the sender to produce a pseudo-random one-time pad of appropriate length from his randomly selected seed. The receiver's ability to extract square roots (based on his trap-door information) allows him to recover the pad and figure out the cleartext.

More formally, let p and q be two randomly chosen distinct large primes congruent to 3 modulo 4, which together form the private key. Their product $n = pq$ is the public key. The cleartext message space is the set of *all* finite binary strings of arbitrary length (any message can thus be directly enciphered without having to break it in pieces and use one of the modes of section 3.5, as was the case with RSA). The randomness space is QR_n , the set of quadratic residues modulo n . The ciphertext message space is the set of pairs formed by a quadratic residue modulo n and a finite binary string.

Let m be some t -bit message. Let x_0 be a random quadratic residue modulo n . Let $BBS_{n,t}(x_0)$ and x_t be defined as in section 4.5. The enciphering of m using seed x_0 and public key n is given by the pair $\langle x_t, m \oplus BBS_{n,t}(x_0) \rangle$ (recall from section 3.5 that “ $\oplus$ ” denotes the bit-wise exclusive-or). Here, $BBS_{n,t}(x_0)$ is used as one-time-pad on cleartext m . The value of x_t is included in the ciphertext in order to allow efficient decipherment by the legitimate receiver, but it is of no help to the eavesdropper. Recall that knowledge of the factors of n is necessary and sufficient for the efficient computation of principal square roots [196]. The *crude* deciphering algorithm consists of computing the pseudo-random sequence *backwards* from x_t using the recurrence equation $x_i = \sqrt{x_{i+1}} \bmod n$. Once $BBS_{n,t}(x_0)$ is thus reconstructed, the plaintext is easily obtained from the ciphertext.

Let l be the number of bits in the modulus n . The efficiency of the enciphering algorithm we have just described is very similar to that of the RSA cryptosystem because it requires one modular *squaring* operation for each bit of the plaintext, whereas RSA would require one modular *exponentiation* for each $(l-1)$ -block of plaintext, and because each exponentiation requires the

computation of l squares and up to l additional multiplications (section 4.1). The crude deciphering algorithm suggested in the previous paragraph is not so good: it requires the computation of one *principal square root* for each bit of plaintext, and each of these is roughly as time consuming as one modular exponentiation. Fortunately, knowledge of the factors of n allows not only random access forwards as described at the end of section 4.5, but also random access backwards. At a cost similar to that of a single exponentiation (or the RSA decipherment of a single block), this allows the legitimate receiver to compute x_0 directly from x_t and then proceed forwards to obtain $BBS_{n,t}(x_0)$ as efficiently as did the sender.

Here is an efficient algorithm for computing x_0 from x_t and the factorization $n = pq$. As a preconditioning step, integers a and b such that $ap + bq = 1$ are computed once and for all by Euclid's extended algorithm. Then, proceed as follows:

```

 $\alpha \leftarrow \text{expo}((p+1)/4, t, p-1)$ 
 $\beta \leftarrow \text{expo}((q+1)/4, t, q-1)$ 
 $u \leftarrow \text{expo}((x_t \bmod p), \alpha, p)$ 
 $v \leftarrow \text{expo}((x_t \bmod q), \beta, q)$ 
return  $(bqu + apv) \bmod n$  .

```

The Blum-Goldwasser probabilistic encryption scheme can be made even faster. A more careful analysis of the BBS pseudo-random generator [231, 232, 5] shows that one can use more than merely the least significant bit after each modular squaring operation. There is no weakening in the resulting pseudo-random sequence if it is made of (approximately) the $\log_2 l$ least significant bits of each x_i . With this improvement, encipherment goes faster than RSA by a factor of about $\log_2 l$ and this is also true for the decipherment of long messages (because it is sufficient to jump backwards only once). In conclusion, recall that not only is this scheme faster than RSA, but the difficulty of breaking it is *proved* equivalent to that of factoring (whereas RSA could be weaker than factoring) and it leaks no partial information on the cleartext if factoring is hard indeed (whereas RSA definitely leaks some partial information, and this may remain so even if random padding is used).

On the other hand, this scheme is totally insecure against a chosen ciphertext attack — we encourage the reader to figure out why.

4.7. Hybrid Systems

Despite all the advantages of public-key cryptography and probabilistic encryption, none of the implementations proposed thus far can compete in *speed* with secret-key systems such as the DES. When a large amount of information has to be communicated, it may be that the use of RSA would be too slow whereas the use of DES would either be impossible (for the lack of shared secret key) or not adequately secure.

The use of a compromise can be valuable in such situations. A *hybrid cryptosystem* uses a public-key cryptosystem once at the outset of the communications in order to share a short piece of information that is then used as key to encipher and decipher the actual message through a secret-key cryptosystem. If the message is sufficiently long, it is preferable to use the public-key cryptosystem several times during the transmission, so as to change secret keys often. Without much slowing down the protocol, this greatly increases the safety of the hybrid system for two reasons: it is easier to break a secret-key system (under a ciphertext-only attack — which is the only type of attack making sense in this context) if much ciphertext is available, and even if the cryptanalyst succeeds at figuring out one of the secret keys, he can only decipher the corresponding part of the message.

APPLICATIONS

Although historically its original driving force, the establishment of secure communication over insecure channels is not the only purpose of cryptography. Indeed, modern cryptography (both secret- and public-key) has innumerable other applications in our information-driven society. In this chapter, we shall discuss some of these in detail, and briefly mention (in section 5.8) several additional applications of a more theoretical nature.

5.1. Authentication

Until now, we have only dealt with the notion of a *passive* cryptanalyst, that is someone whose purpose is merely to eavesdrop on the communication channel. An *active* cryptanalyst (also known as a *tamperer*) goes further: not content with listening on the communications channel, he may also inject messages of his own in the hope of deceiving the receiver into believing it was sent by someone else. Needless to say that financial transactions, for instance, must be protected against such tampering perhaps even more so than they need privacy.

The purpose of an *authentication scheme* is to detect the presence of such an intruder. Whenever B receives a message that claims to have been sent by A, the scheme must allow him to ascertain not only that the message originated from A, but that it was not modified in transit. We allow the tamperer to eavesdrop on as many authenticated messages as he wishes, and his goal is to come up with one fake message that will escape detection. This fake message could either be completely different from those already intercepted, it could be only minimally different from one of them, or it could be anything between these two extremes. In order to detect replays, it is important that each message include a time stamp or a sequence number.

It is often believed that any secret-key cryptosystem can be used directly for the simultaneous purposes of privacy and authentication. Let k be the

secret key shared between A and B, and let E_k and D_k be the corresponding enciphering and deciphering algorithms. Should B receive from A some ciphertext c , he would decipher it as $m = D_k(c)$. If the resulting cleartext m makes sense, B might feel confident that it was indeed enciphered as $c = E_k(m)$ by the only other person knowing the secret key k , namely A. The reasoning is that a tamperer unaware of k would be incapable of making up fake ciphertexts that would not decipher into meaningless jumbles. *This belief is false* because the tamperer may possess known pairs of corresponding plaintext/ciphertext, and he may be capable of patching pieces of the ciphertexts together in a meaningful way. Even the best cryptosystems can be totally worthless for the purposes of authenticity, particularly if used in the Electronic Code Book mode (section 3.5).

In order to see why this is so, let us consider a cryptosystem that offers Shannon's perfect secrecy. Assume an automatic teller machine uses a one-time pad to transmit information to the main bank computer, and assume the plaintext information is sent in a format known to the tamperer. The latter can then go to the teller, deposit ten dollars, intercept the encrypted message on its way to the computer, deduce the actual pad used from the encrypted message and his knowledge of what the cleartext should be, and modify the relevant positions in the ciphertext to change the amount to whatever he pleases. Even if the cleartext is not known precisely, it is enough to suspect its format in order to deduce how tampering can be attempted without running too much risk of detection.

An *authentication scheme* consists of a *key space* K and, for each $k \in K$, of a *message space* M_k , a *tag space* T_k and an *authentication function* $A_k : M_k \rightarrow T_k$. Given any key k , it must be easy to obtain an efficient algorithm for computing A_k . The authentication scheme is used as follows. If A and B expect that they might eventually have to authenticate messages between them, they must initially agree on some secret key $k \in K$. Whenever A wishes to authenticate some message $m \in M_k$ for B, she computes $t = A_k(m)$ and sends it along together with m . In order to verify the authenticity of the message, B also computes $A_k(m)$ and compares it with the tag he received.

Of course, this does not preclude the encryption of m if both privacy and authenticity are sought.

There is no need for the authentication function to be one-to-one and the tag space can be substantially smaller than the message space. It should not be too small, however, so that a random tag has but a negligible probability of being correct for the lucky tamperer's chosen message.

As was the case for secret-key cryptography, one can distinguish different levels of attack:

- *Known message attack*: the tamperer has eavesdropped on several authenticated pairs $\langle m_1, t_1 \rangle, \langle m_2, t_2 \rangle, \dots, \langle m_i, t_i \rangle$ such that $t_j = A_k(m_j)$ for each $1 \leq j \leq i$ and for an unknown k ; he is to:
 - figure out k or, lacking this ability,
 - figure out $t_{i+1} = A_k(m_{i+1})$ for a message m_{i+1} of his choice or, lacking this ability,
 - figure out $t_{i+1} = A_k(m_{i+1})$ for *any* m_{i+1} distinct from the m_j 's, $1 \leq j \leq i$ (in the hope of creating confusion) or, if even this is not possible,
 - figure out some pair $\langle m_{i+1}, t_{i+1} \rangle$ for a new m_{i+1} such that t_{i+1} has more than a negligible probability of being the correct tag $A_k(m_{i+1})$;
- *Chosen message attack*: the tamperer initially gets to choose messages $m_1, m_2, \dots, m_i$ and he is given the corresponding tags $t_1 = A_k(m_1), t_2 = A_k(m_2), \dots, t_i = A_k(m_i)$; he is then to perform one of the tasks mentioned above (this type of attack could be mounted, for instance, by a temporary bank employee having access to the authentication black box machine for a limited time).

Despite what was said previously, secret-key cryptosystems can often be used as a building block in authentication schemes. This is possible if the cryptosystem has a fixed block size and if the message to be authenticated is long enough to span several blocks. The idea is to encipher the message under either the Cipher Block Chaining or the Cipher Feedback mode and to release as authentication tag only the last block thus produced. By definition

of these modes, the tag will depend on the entire message. If both privacy and authenticity are sought and if the same secret-key cryptosystem is used for both functions, it is much safer to use two *distinct* secret keys [152].

When a secret-key cryptosystem is used for privacy considerations, safety is substantially increased by changing keys often. Somewhat curiously, this is not the case for authentication purposes [97]. Furthermore, the consequences of an occasional forged tag escaping detection are likely to be much more dramatic than that of an occasional breach in secrecy brought about by a successful decryption by an enemy cryptanalyst. As a consequence, a careful choice of which cryptosystem should be used is more crucial for authentication than for privacy.

Combining the notions of one-time pad and *universal hashing* [63], Wegman and Carter have developed authentication tags of an entirely different nature: they propose an information-theoretically unbreakable authentication scheme [236]. As is the case with the classic use of one-time pads, the amount of necessary secret information is proportional to the numbers of messages that have to be authenticated (although it is independent of their lengths). In other words, some bits of the secret key are used up each time a message is authenticated. If the users are willing to use up t bits per message, Wegman and Carter have shown that even an enemy with unlimited computing power making use of a chosen message attack cannot forge a false tag with a probability of success better than 2^{-t} . They also prove that this is optimal.

Brassard has shown how to combine Wegman-Carter tags with the BBS pseudo-random generator (section 4.5) in order to make do with a fixed short key independent of the number of messages. The resulting scheme is however secure only against feasible computations (not unlimited computing power) and only if factoring is hard [48]. Goldreich, Goldwasser and Micali have also proposed a computationally secure authentication scheme [128] as an application of their notion of poly-random collection of functions [129] (section 5.8).

5.2. Digital Signature

Although an authentication scheme allows A to gain high confidence that the message she received originated from B, it does not allow her to convince anyone else that B actually sent it. This makes authentication schemes suitable between two mutually trusting parties, but it cannot be used to settle a dispute between them. The advent of trap-door one-way functions has made possible the stronger notion of *digital signature*. If B sends a digitally signed message to A, the latter will not only be convinced that the message was signed by none but the former, but she will also be able to *prove to a judge* that B actually signed that very message. This notion was invented by Diffie and Hellman [101]; read also [205, 195, 196, 176, 136]. Digital signature and electronic mail offer substantial advantages over traditional paperwork. As we shall mention in section 5.8, they even make possible electronic certified mail and electronic signing of contracts.

It should be clear why an authentication scheme does not provide digital signature. If A authenticates a message for B in the way described in section 5.1, it would have been just as easy for B to come up with the corresponding tag himself. The only reason B trusts that the computation was performed by A is that he knows he did not do it; of course, there would be no such evidence for a judge.

A public-key cryptosystem offers *signature capability* if, for each key k , $M_k = C_k$ and if $E_k(D_k(m)) = m$ for every message $m \in M_k$ (the second condition is a consequence of the first whenever M_k is a finite set). It is used as follows for signature purposes. Let a be some user A's private key and let E_a and D_a be her enciphering and deciphering functions. If the cryptosystem is secure, only A can compute D_a efficiently but everyone knows how to compute E_a .

Consider now some cleartext message m and let $s = D_a(m)$. Anyone can compute $E_a(s)$ and find out that its value is m . Only A, however, has the knowledge to come up with some s such that $E_a(s) = m$. In this sense, s can be considered as A's signature of message m . If B shows s to a judge and if $E_a(s) =$ "I owe B one thousand dollars", the judge should be convinced that no one but A could have signed this statement. In other words, the private

deciphering algorithm can be thought of as a signature algorithm and the public deciphering algorithm can be thought of as the corresponding signature verification algorithm.

Digital signature can be used in conjunction with public-key encryption if secrecy is also desirable. Let b be user B's private key. If A wants to send signed message m privately to B, she uses her private signature algorithm D_a and *his* public enciphering algorithm E_b to produce $c = E_b(D_a(m))$. If she then sends c to B over an insecure channel, the latter can compute A's signature of m as $s = D_b(c)$ and then recover the cleartext message as $m = E_a(s)$. (This assumes of course that the message header says in the clear that it originates from A, so that B knows whose signature verification to apply in order to obtain the plaintext.) Moreover, if B keeps s , he can prove to a judge that A sent him message m , as we explained previously.

The natural alternative solution would be for A to send $\hat{c} = D_a(E_b(m))$, allowing B to decipher as $m = D_b(E_a(\hat{c}))$. This is to be avoided because it would allow a tamperer T (with private key t) to intercept $\hat{c}$ before it reaches B. If T then computed $\bar{c} = D_t(E_a(\hat{c}))$, he could forward $\bar{c}$ to B, pretending it comes from him. Unsuspecting, B would then compute $D_b(E_t(\bar{c})) = m$ and be convinced message m originated from T (unless of course m starts with "My name is A"). The point is that it allows T to actually sign a message whose contents he is unable to read. Although it is hard to imagine why T might wish to do this, it is better to avoid the possibility altogether.

If RSA is used for both privacy and signature, it may be preferable that each user keeps two distinct pairs of trap-door functions for the two distinct purposes. Each user would therefore have one entry in the public enciphering directory and another entry in the public signature verification directory. This separation serves two purposes. Firstly, it helps avoid the re-blocking problem that would otherwise occur when the sender's modulus is larger than the receiver's [205]. Secondly, we have already seen (section 4.4) that RSA is weak under chosen ciphertext attacks. Such attacks can be harder to mount if signing is different from deciphering.

Care must be taken when one says that B's mere knowledge of s such that $E_a(s) = m$ is to be considered as proof that A must have signed m . After all,

B could select some random s , compute $m = E_a(s)$ — which is almost certain to make no sense at all — and then pretend A must be mad for having signed such non-sense as m ! Things get worse if the density of meaningful messages is high (such as some types of numerical data rather than plaintext English) for then B could choose several s 's at random until he is lucky enough to hit upon an s such that $E_a(s)$ is meaningful. For these reasons, it is well-advised to agree on some sort of *normal form*, and to accept s as signature for m only if $m = E_a(s)$ and m is in the normal form. Unfortunately, it is not yet clear what a safe normal form could be with RSA. For instance, padding with a fixed number of zeroes is not safe [151].

The RSA cryptosystem offers signature capability. Using it presents drawbacks similar to those it had as a public-key encryption scheme. In particular, it is not known to be as hard to break as it is to factor large integers. Even if it were genuinely hard, given a chosen plaintext m and someone else's public key $\langle e, n \rangle$, to compute the signature s such that $s^e \bmod n = m$, it may be much easier to do so given also $\langle \hat{s}, \hat{m} \rangle$, where $\hat{s}$ is the legitimate user's signature of some message $\hat{m}$ only slightly different from m . In other words, it could become easier to forge signatures for a given message after having seen the proper signatures of several similar messages.

The above paragraph suggests one should use probabilistic encryption rather than public-key cryptography in order to design digital signature schemes. This hope turns out to be ill-founded: the Blum-Goldwasser probabilistic encryption scheme is as worthless for signature as the one-time pad was worthless for authentication. Fortunately, strong signature schemes have been developed with a proof (under the assumption that factoring is hard) that they withstand even a chosen message attack (the definition given in section 5.1 is easy to adapt for the case of signatures). These schemes are rather complicated and we shall not describe them here [137, 135, 126]. For an extensive discussion of digital signature, its problems and some solutions, consult [135]. Read also [211].

5.3. User Identification

(Written with Claude Crépeau and Claude Goutier)

The basic question is the following: whenever a would-be user seeks the services of a computer (or when a customer wants to use an automatic teller, or when the teller wishes to access the bank's central computer), how can the latter make sure the former is not forging a false identity? The classic solution to this problem is through the use of passwords. Assuming the user and the computer share some confidential piece of information, the latter can request this information from the former whenever his identity is questionable. Clearly, the value of such a scheme depends crucially on the ability to keep passwords secret, which points at two weak links: the existence of a password table somewhere in the computer and the threat of wire taps.

The mere fact that the passwords are available to the computer is very dangerous. No matter how well they are hidden or protected, a clever enemy can always find them wherever they are (preferably, but not necessarily, if he has physical access to the computer, its secondary storage and the operating systems listing). Moreover, in most computer environments, there is at least one person, the System Administrator or Super User, who has legal access to the password directory. How much trust can other users put in this person? At the very least, prudent users should use distinct passwords on different systems.

This threat can easily be overcome in practice [106, 192, 238, etc.]. The key idea is that the computer does not really need to *know* the actual passwords; he merely needs to be capable of *validating* given passwords. This can be implemented through the use of one-way functions. It is enough for the computer to store the *images* of the users' passwords under some fixed one-way function. Because it is one-way, there is no point in keeping the function itself secret. Whenever a user wishes to prove his identity, he transmits his actual password on which the computer immediately applies the one-way function. The result can then be validated against the computer's table. Notice that an enemy would find useless a listing of the (transformed) password table because of his inability to reverse the one-way function, hence to determine the actual passwords.

This scheme can be rather weak if the users are likely to use easy-to-guess passwords. The following attack can often be mounted successfully by someone who has access to the table of (transformed) passwords and to the algorithm for computing the one-way function. Choose the one thousand (say) most common passwords, compute the one-way function on them, and efficiently compare the results with the encrypted table (making use of hashing or sorting techniques). Each hit yields some user's password. This attack does not allow one to choose whose passwords he is seeking, but it is very effective at finding several of them.

Although it does nothing additional towards protecting individual passwords, *salting* is a technique that nearly eliminates the above collective threat. The system chooses a random string for each user. The password table associates with each username the corresponding random string (in the clear) together with the value of the one-way function applied on the *concatenation* of the user's actual password with his random string. As a result, two different users who may happen to have chosen the same password will have distinct entries in the system's table.

Despite this improvement, there remains one major weakness. Every time a user wishes to establish his identity, he must transmit his actual password to the computer. The password is then vulnerable to interception either as it goes through the communications link to the computer or inside the computer memory before it has been irreversibly transformed by the one-way function.

To get rid of this threat, it must be that not only the computer does not have the passwords in storage, but it is *never* told them under any circumstances. This seemingly paradoxical requirement can be satisfied through the combination of a World War II idea — the identification-friend-or-foe (I.F.F.) systems [154] — with the notion of trap-door one-way functions. It requires users to have the availability of their own intelligent terminals, or preferably of personal “smart” cards [178, 74].

In order to solve the user identification problem, each user of the system chooses a private random key from some public-key cryptosystem. He uses it to generate a corresponding pair of enciphering and deciphering algorithms. He reveals the enciphering algorithm to the computer and he programs the

deciphering algorithm into his terminal. Whenever the computer wishes to assert the identity of a user, it generates a random message, computes the alleged user's enciphering function on that input and sends the result to the user's terminal as a challenge. The latter runs its secret program to recompute the original message, which is sent back to the computer for validation. It follows immediately from the definition of trap-door one-way functions that only the legitimate user is capable of meeting the challenges efficiently, even should his enciphering function (as programmed inside the computer) be known to an enemy. As promised, the user's password, which is in fact his efficient deciphering algorithm, is never revealed to the computer.

Notice that it does no good for an eavesdropper to monitor the challenges and responses exchanged between the computer and a given user because he could have produced the same exchange (probabilistically speaking) just from knowing the user's public information. Moreover, should he try to impersonate this user, he is nearly certain to face challenges different from those he had monitored. On the other hand, the user should not trust the computer for selecting the challenges truly at random, hence the underlying cryptographic scheme should be safe against chosen ciphertext attacks (section 4.3). The Blum-Goldwasser probabilistic encryption scheme (section 4.6) should therefore *not* be used.

We do not really need the full strength of trap-door one-way functions in order to get a working user identification scheme. Assume for instance that deciphering can be done in less than one second with the availability of the private key, but that it can also be performed in a few hours without it. Such fast cryptanalysis would spell doom for cryptographic purposes. However, it would still be good as a user identification scheme because the computer can monitor how long it takes the user's terminal to reply. A correct answer computed too late would be just as useless for the intruder as no answer at all.

The scheme we have just described is not really a *user* identification scheme but rather a *terminal* identification scheme. Any person having physical access to the terminal can use the computer, even if the terminal access is not legitimate. One solution is to require that the user identifies itself to his

own terminal before using it. This can be done through a classical password scheme as long as both the user's password and the terminal's deciphering algorithm are kept in a black box that self-destructs if tampered with. A better idea is for the deciphering algorithm to be computed by the user's personal smart card [178, 74]. With this solution, the terminals would need minimal intelligence and they could all be identical, allowing the user to access the system from any physical location equipped with the proper terminals. Important issues concerning the dangers of using smart cards are mentioned in [123]; it would be much safer if the "card" had its own display and keyboard.

This identification scheme can be used in a *continuous challenge* mode, by which we mean that the computer can challenge the terminal at regular intervals. This removes the further threat of an intruder that would wait for the legitimate user to establish connection before diverting the communications line to his own terminal (or better still to avoid raising suspicion: wait until the legitimate user sends the "logout" command before snatching the line away from him). If additional calculations and delays are tolerable, it would be even safer to authenticate every single request coming from the user.

For a practical implementation, it is best if the challenge can be *constructed* very quickly even if *meeting* the challenge requires more computation. This is so because the mainframe may have a large number of terminals to attend, whereas each individual terminal can work on its challenges in the background (while still functioning at full speed from its user's point of view). For instance, one could use the principle of RSA but with public exponent systematically set to 3 (the known weaknesses of using a small enciphering exponent [143] do not apply here at all).

Of course, the problem of controlling someone's identity is not limited to restricting the use of a computer. In general, the identification problem can be solved using *minimum disclosure techniques*, a notion discussed in section 5.6. For more detail on minimum disclosure (*zero-knowledge*) identification protocols, read [112, 109], but be aware of the potential frauds described in [16].

5.4. Coin Flipping

At the 1982 *IEEE Computer Conference (CompCon)*, Manuel Blum gave a seminal talk titled “Coin flipping by telephone: A protocol for solving impossible problems” [35]. This was but the beginning of a fascinating story, as more and more “impossible problems” have since been solved. For a review of some of the most outstanding recent cryptologic achievements made possible by academic research, consult [50]. Blum introduced the coin flipping problem with the following delightful story: “Alice and Bob want to flip a coin by telephone. (They have just divorced, live in different cities, want to decide who gets the car.) Bob would not like to tell Alice HEADS and hear Alice (at the other end of the line) say ‘Here goes... I’m flipping the coin.... You lost!’”.

The trouble with this naive protocol, of course, is that it allows Alice to choose the coin “toss” after Bob told her his guess. In order to prevent this kind of cheating, Bob should request that Alice toss the coin first and be committed to its outcome. Only then should Bob announce his guess. There are two requirements for this idea to lead to a viable protocol: Alice must be incapable of flipping the coin over after hearing Bob’s guess and Bob must be incapable of knowing how the coin landed before announcing his guess.

Let $f: X \rightarrow Y$ be a one-way function and assume X is a finite set of integers containing exactly as many even as odd numbers. Assuming Alice and Bob have agreed on the function f , the following protocol was originally suggested by Blum and Micali:

Alice chooses a random element $x \in X$,
 computes $y = f(x)$,
 announces y to Bob ;

Bob guesses whether x is even or odd,
 announces his guess to Alice ;

Alice tells Bob whether he guessed right or not,
 proves it to him by disclosing x ;

Bob verifies Alice’s good faith by checking that $f(x) = y$.

The intuition is that Alice is committed to x because she gave $y = f(x)$ to Bob, but Bob cannot figure out x for himself because f is one-way. (Section 5.5 comes back in more detail on the notion of *bit commitment*.) This intuition is good but it leaves the door open for both Alice and Bob to cheat if the function f is not chosen with enough care (even if it is genuinely one-way).

If f is not one-to-one, it is possible that Alice knows x and $\hat{x}$ such that $f(x) = f(\hat{x})$ and $x + \hat{x}$ is odd (this is not precluded by the definition of one-wayness). In this case, Alice would announce $y = f(x)$ without being in fact committed to either x or $\hat{x}$. If this were so, Alice could cheat. Conversely, it could be that f is one-to-one and that it is infeasible to compute x from $f(x)$, yet this does not necessarily prevent the *parity* of x from being easy to figure out upon seeing $f(x)$. Even if the parity of x is hard to determine with certainty given $f(x)$, it could still be that it can be guessed with a better than 50% chance of success. In either case, Bob could cheat.

Under the assumption that factoring large integers is infeasible, these potential difficulties can be overcome by the following protocol.

Alice randomly chooses a Blum integer n and some $x \in \mathbb{Z}_n^*$,
 computes $y = x^2 \bmod n$ and $z = y^2 \bmod n$,
 discloses n and z to Bob;

Bob announces whether he thinks y is even or odd;

Alice discloses x and y to Bob,
 convinces Bob that n is a Blum integer;

Bob verifies that $y = x^2 \bmod n$ and $z = y^2 \bmod n$.

In other words, Alice gives Bob a quadratic residue z modulo a Blum integer n and Bob is to decide whether its principal square root y is even or odd. As we have already seen (section 4.5), Bob cannot do any better than toss a coin to make his guess. On the other hand, it is crucial that n be a Blum integer for otherwise it could be that z has two square roots of different parity, both being quadratic residues. This is why Alice must “convince Bob that n is a Blum integer”. For this, Alice could give Bob the factoring of n (it would be too late by then for Bob to use this information in order to

influence the coin toss). If Alice wishes to use her Blum integer n for other purposes, or use it for several coin tosses, she could use a minimum disclosure interactive protocol to convince Bob of her good faith [140] — see section 5.6.

An additional benefit of this protocol is that it allows coin flipping *in a well*. This means that there comes a point in the protocol when Alice knows the outcome and cannot change it, but she can delay at her will disclosing the result to Bob. The analogy is with Bob flipping a physical coin in a deep well close to Alice: she can see the coin but not change it, and Bob is too far to see it but Alice may eventually allow him to come closer and look into the well.

A protocol based on quantum physics for coin flipping in the well is described in [18]. It remains secure even if either or both parties have unlimited computing power. Although one of the parties can cheat in theory, this would require a technology far beyond that in any foreseeable future.

5.5. Bit Commitment Schemes

(Written with Claude Crépeau and David Chaum)

The notion of *bit commitment* is a powerful and general primitive for the establishment of several cryptoprotocols. The purpose of bit commitment is to allow Alice to commit to the value of a bit in a way that prevents Bob from learning it without her help, but also in a way that prevents her from changing the value of this bit. If a secure bit commitment scheme is available, coin flipping (as described in section 5.4) becomes trivial to implement: Alice commits to the value of a randomly chosen bit, Bob guesses this value and announces his guess, and Alice shows him whether he had guessed correctly. A more elaborate application of bit commitment schemes is described in the next section.

Bit commitment is implemented through a primitive, which we call for convenience the “*blob*”. Each blob is used as a commitment to either 0 or 1. For the sake of generality, we do not impose any restriction on the nature of blobs — they could be made out of fairy dust if this were useful. By “Alice commits to a blob”, we mean that Alice has a blob “in mind” and that she does something that will force her to stick to this blob in the future. If the

blob itself can be represented as a bit string — as in most practical cases — committing to a blob can be as simple as showing it in the clear. The abstract defining properties of blobs are as follow:

- i) Alice can commit to blobs: by committing to a blob, she is in effect committing to a bit.
- ii) Alice can *open* any blob she has committed to: she can convince Bob of the value of the bit she in effect committed to when she committed to the blob. Thus, there is no blob she is able to “open” both as 0 and as 1.
- iii) Bob cannot learn anything about which way Alice is able to open any unopened blob she has committed to. This remains true even after other blobs have been opened by Alice.
- iv) Blobs do not carry “side information”: the blobs themselves as well as the processes by which Alice commits to and opens them are uncorrelated to any other secret she might wish to keep from Bob.

Consider the following illustrative implementation of a blob. When Alice wishes to commit to a bit (property (i)), she writes it on the floor and, before allowing Bob to look, she covers it with opaque tape. Although Bob cannot tell which bit is hidden under the tape (property (iii)), Alice can no longer change it. To “open the blob” (property (ii)), Alice allows Bob to strip off the tape and look at the bit. Property (iv) is satisfied provided the way in which the bit is written on the floor, the tape and its placement are all uncorrelated to any secret Alice might wish to keep from Bob.

If the blobs are to be used inside a conventional cryptoprotocol that proceeds by exchange of digital messages, they should be digitally encoded themselves. In this case, there is an apparent contradiction between the properties required of the blobs. Property (ii) implies that blobs that Alice can open as 0 be different from those she can open as 1. But then, what is it that prevents Bob from detecting this difference and thus violate property (iii)?

One possible answer is to ask that it is not the blob itself that determines a bit, but rather Alice’s knowledge about it. If this idea is implemented carefully, it becomes impossible for Bob to learn anything about which way Alice is able to open any unopened blob she has committed to (in the strongest

information theoretic sense), but additional knowledge could in principle allow Alice to violate property (ii). On the other hand, if we insist that property (ii) should be unconditionally enforced, a dual solution is to require that blobs used by Alice as commitments to 0 be distinct from those she uses to commit to 1. In this case, she is irrevocably committed to a specific bit each time she commits to a blob, but as a consequence Bob could in principle violate property (iii). Although either approach makes it conceivable for one of the parties to cheat, this can be made computationally infeasible under suitable cryptographic assumptions.

As an elementary (but probably not very secure) example, consider two isomorphic graphs G and H upon which Alice and Bob agree. Assume that Alice is convinced that they are isomorphic, but that she does not actually know an isomorphism between them. Suppose further that she is computationally incapable of finding such an isomorphism in a reasonable amount of time. (Let us postpone until section 5.6 the question of how Alice could be convinced that the graphs are isomorphic, without explicitly knowing an isomorphism.) In this setting, Alice agrees with Bob that any graph for which she can show an isomorphism with G (resp. H) is a commitment to the bit 0 (resp. 1). Referring to the defining properties of blobs, property (i) holds because Alice can commit to the bit 0 (resp. 1) by randomly permuting the vertices of G (resp. H) and showing the resulting graph — the blob — to Bob. In order to open a blob, it suffices for Alice to show Bob the isomorphism she knows with G or H , whichever is the case. Property (ii) holds if and only if Alice cannot find an isomorphism between G and H . (More precisely, in order for Alice to break property (ii), she must have obtained information that makes it easy for her to discover such an isomorphism.) Property (iii) holds unconditionally because blobs used by Alice as commitments to 0 are information theoretically indistinguishable from those used as commitments to 1: they are random isomorphic copies of G (and thus of H). Finally, property (iv) is satisfied because Alice permutes the vertices of G randomly, thus in a way uncorrelated with anything else.

This example illustrates the case in which it is not the blob itself (some graph isomorphic to both G and H) that determines a bit, but rather Alice's

knowledge about it (the actual isomorphism known by Alice between this graph and either G or H). A more practical but slightly more technical example of the same type of bit commitment scheme is based on the intractability assumption of the discrete logarithm problem (see section 4.1) [71, 44]. Let p be a large prime and let α generate $\mathbb{Z}_p^*$, the multiplicative group of integers modulo p . Given any integer y , recall that it is easy to compute $\alpha^y \bmod p$, but no efficient algorithm is known to invert this process. Let us furthermore assume that it remains infeasible to compute y from α , p and $\alpha^y \bmod p$ even if the factorization of $p - 1$ is known.

Initially, Alice and Bob agree on a suitable prime number p for which both of them know the factorization of $p - 1$. They also agree on α , a generator of $\mathbb{Z}_p^*$. Thanks to their knowledge of the factors of $p - 1$, they can both verify with certainty that p is a prime and that α is a generator [158]. These same parameters p and α can be public, in the sense that they can be used by all parties wishing to engage in bit commitment protocols. At the outset, Bob also chooses a random $s \in \mathbb{Z}_p^*$ ($s \neq 1$) and gives it to Peggy. According to our assumption, Alice cannot compute e such that $s \equiv \alpha^e \pmod{p}$.

In order to commit to some bit b , Alice chooses a y randomly between 0 and $p - 2$ and she computes $x = s^b \alpha^y \bmod p$. She gives x — the blob — to Bob but keeps y secret as her *witness* that allows her to open x as bit b . Clearly, any element of $\mathbb{Z}_p^*$ can be used by Alice as commitment to 0 just as well as to 1, depending only on her knowledge about it. Moreover, elements of $\mathbb{Z}_p^*$ are obtained by this process with a uniform probability distribution, independently of the bit Alice wishes to commit to. Therefore, property (iii) holds unconditionally here as well: blobs committed to by Alice contain no information on the way in which she could open them. Property (ii) holds computationally because Alice could easily obtain e (which we assumed to be infeasible for her) from knowledge of y_0 and y_1 such that $\alpha^{y_1} \equiv s \alpha^{y_2} \pmod{p}$. Property (iv) holds because Alice chooses y randomly.

The intractability assumption of the discrete logarithm can also be used to implement a bit commitment scheme of the dual kind [53]. Again, let p be a large prime and let α generate $\mathbb{Z}_p^*$. Let u be the smallest integer such that 2^u does not divide $p - 1$. Given any $s \in \mathbb{Z}_p^*$, let $hard(s)$ denote the u th least

significant bit of the unique integer e such that $0 \leq e \leq p-2$ and $s \equiv \alpha^e \pmod{p}$. Under the intractability assumption of the discrete logarithm, it is infeasible to learn anything about $\text{hard}(s)$ (given only p , α and a randomly selected s) because this problem has been proved to be as hard as that of the discrete logarithm itself (in a strong probabilistic sense) [189]. (The $u-1$ least significant bits of e are easy to compute given s .)

Initially, Alice and Bob agree on p and α exactly as above. In order to commit to bit b , Alice chooses an integer y between 0 and $p-2$ at random, except that she sets the u th least significant bit of y to b . She computes the blob $x = \alpha^y \bmod p$ and gives it to Bob. As for the two previous implementations, it is clear that the defining blob properties (i) and (iv) hold. However, property (ii) now holds unconditionally because α is a generator of $\mathbb{Z}_p^*$, hence the discrete logarithm of x is uniquely defined, and so is its u th least significant bit. Property (iii) holds from a computational point of view under our cryptographic assumption, since the bit encoded by blob s is precisely $\text{hard}(s)$.

Several other implementations of bit commitment schemes are described in [53]. Let us also mention another implementation that remains secure even if both Alice and Bob have unlimited computing power, but that relies on dogmas of quantum physics [22]. Yet another implementation is unconditionally secure for all parties considered in a *multi-party* environment (see section 5.8) under the assumption that no more than one-third of the participants are dishonest [70].

5.6. Minimum Disclosure Proofs

(Written with Claude Crépeau and David Chaum)

Assume Peggy (“the Prover”) knows some information. For instance, this could be the proof of a theorem or the prime factorization of a large integer. Assume further that Peggy’s information is *verifiable*, in the sense that there exists an efficient procedure capable of certifying its validity. In order to convince Vic (“the Verifier”) of this fact, Peggy could simply reveal the information to him so that he could perform the certifying procedure himself. This would be a *maximum disclosure* proof, since it results in Vic learning all the

information. He could therefore later show it to someone else and even claim it to have been his originally.

In this section we describe a general and efficient protocol for obtaining *minimum disclosure* proofs. This protocol allows Peggy to convince Vic, beyond any reasonable doubt, that she has information that would pass the certifying procedure, but in a way that does not help him determine this information. For example, if Peggy’s information is the proof of a theorem, Vic is left with the conviction that Peggy knows how to prove it, and hence that the theorem is true. However, Vic is not given even a clue as to how the proof might proceed (except perhaps for an upper limit on its length). Although Peggy’s original information is verifiable, the conviction thus obtained by Vic may not be. In particular, conducting the protocol with Peggy need not (and in many cases *will* not) enable Vic to subsequently convince someone else.

In general this is achieved by an interactive protocol consisting of several rounds. In each round, Peggy allows Vic to ask her one of two possible challenges in a way that she could answer both of them only if she had the secret she claims to have, but also in a way that answering either one challenge does not reveal any information about this secret. Because Peggy cannot predict ahead of time which challenges Vic will issue, each round increases his confidence in her claim. In fact, she could only hope to fool him in k successive rounds with exponentially vanishing probability 2^{-k} . We call such techniques *cut-and-choose* because each round is similar to the classic “protocol” by which two children split a piece of cake — one of them cuts and the other one chooses. The great utility of such cut-and-choose is that it gives an exponential increase in security at the cost of only a linear increase in the number of rounds.

The earliest use of cut-and-choose we know of in the context of cryptographic interactive protocols was presented by Rabin in 1977 [195]. The concept of *interactive protocol* was subsequently formalized and the notion of *zero-knowledge* was introduced by Goldwasser, Micali and Rackoff [134]. (Zero-knowledge is similar to minimum disclosure, except that it requires that Vic should not obtain *anything* at all from the protocol beyond learning that

Peggy has the information she claims to have.) Also, Babai formalized a notion similar to that of interactive protocols [8]. The models proposed in [134, 8] assume that the prover has unlimited computing power. A different model was set forward by Chaum, which emphasizes the unconditional privacy of the prover's secret information [66], even if the verifier has unlimited computing resources. We shall concentrate here on a model in which all parties involved are assumed to have “reasonable” computing power.

A nice and simple example of minimum disclosure proof allows Peggy to convince Vic that she knows an isomorphism between two given graphs G and H , in a way that does not help Vic figure out this isomorphism [130]. This example illustrates the principle of cut-and-choose (and ties up a loose end in section 5.5 about the example of bit commitment scheme based on graph isomorphism). In one round of the protocol, Peggy permutes randomly the vertices of G to produce some graph K isomorphic to both graphs G and H , and she gives K to Vic. If Peggy is honest, she knows an isomorphism between G and K (because she created it) and one between H and K (by composition with the isomorphism she claims to know between G and H).

At this point, Vic chooses either G or H , and challenges Peggy to produce an isomorphism between his chosen graph and K . Clearly, Peggy is capable to meet this challenge if she is honest. However, no matter how she proceeded to create K (assuming she is trying to cheat), she cannot be ready to answer both possible challenges unless she actually knows an isomorphism between G and H . Therefore, since she cannot predict which challenge Vic will issue, any attempted cheating will be detected by Vic with probability at least one-half. The probability of undetected cheating can thus be made arbitrarily small by repeating this process a sufficient number of times. It is more subtle to see that this protocol cannot help Vic figure out an isomorphism between G and H . For more detail, consult [130].

The above protocol works well for the specific problem of graph isomorphism. However, In order to show that *any* efficiently verifiable information can be handled by a minimum disclosure proof, we need a more powerful result. By Cook's theorem [77], a minimum disclosure proof technique for satisfiability (or any other **NP**-complete problem [121]) would automatically

yield one for any positive statement concerning a language in **NP**. Under suitable cryptographic assumptions, such protocols were obtained by Goldreich, Micali and Wigderson (for graph 3-colourability) [130] and Chaum (for satisfiability) [68]. A similar result was obtained independently but subsequently by Brassard and Crépeau [55, 56]. The notion of minimum disclosure proof extends in a natural way to *probabilistically* verifiable information [53]. It is also possible to *publish* minimum disclosure proofs (thus interaction is not necessary) [39]. Read also [117, 38, 73, 14, 140, 227, 228, 54]. The controversy about this topic in relation to the United States Department of Defense is reported in [124, 168]. We now give Chaum’s construction in detail.

Assume Peggy knows a satisfying assignment of truth values for some Boolean formula. The protocol described below allows Peggy to convince Vic that she knows such an assignment without revealing any information about it. (The formula itself is not a secret; it is only the satisfying assignment that Peggy wishes to conceal from Vic.) As an example, consider the Boolean formula

$$\Psi = [(p \text{ and } q) \text{ xor } (\bar{q} \text{ or } r)] \text{ and } [(\bar{r} \text{ xor } q) \text{ or } (p \text{ and } \bar{r})]$$

and let $\langle p = \text{true}, q = \text{false}, r = \text{true} \rangle$ be Peggy’s secret satisfying assignment. (Of course, this is a toy example since it would be too easy for Vic to find out how to satisfy such a simple Boolean expression.)

As a first step, Peggy and Vic agree on the layout of a Boolean circuit to compute Ψ . For simplicity, we use only basic binary gates and negations. The circuit for Ψ is illustrated in figure 1. In addition, this figure shows Peggy’s satisfying assignment and the truth table of each gate (except the negation gates). Observe that one row is outlined in each truth table, corresponding to the circuit’s computation on Peggy’s satisfying assignment. Seeing the rows outlined is enough to easily verify that Ψ is satisfiable. This can be achieved by simple independent checks on the consistency of each wire. For instance, the output of the top left “AND” gate is 0, which is indeed the first input of the middle row “EXCLUSIVE-OR” gate. Also, the first input to the top left and top right “AND” gates is the same, as it should be since they correspond to the same input variable. Finally, the output of

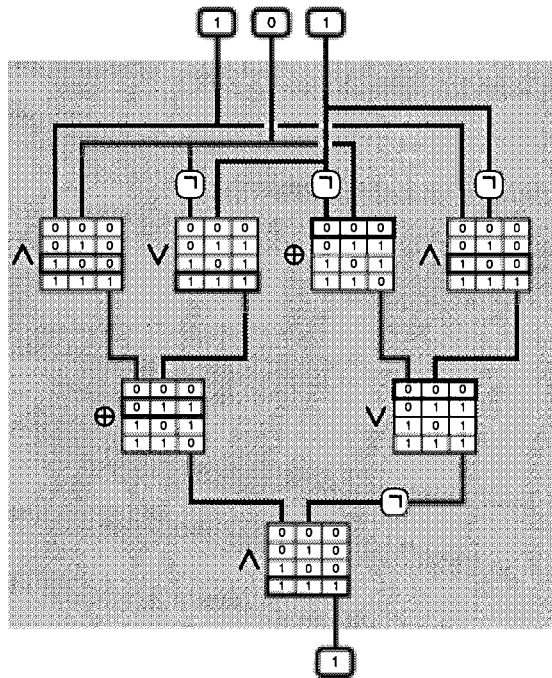


Figure 1. A Boolean circuit with explicit truth tables and rows outlined.

the final gate is 1. Notice that seeing these outlined rows also gives away the corresponding satisfying assignment (even if it were not written explicitly). We want a protocol that allows Peggy to convince Vic that she knows how to so outline one row in each truth table, but without revealing any information about which rows they are.

Once again, this is achieved by an interactive protocol consisting of several rounds. In each round, Peggy “scrambles” the circuit’s truth tables and commits to a corresponding collection of blobs (see section 5.5). At this point, Vic issues one of two possible challenges to Peggy: one challenge requires Peggy to show that the blobs really encode a valid scrambling of the circuit’s truth tables; the other challenge requires Peggy to open the rows that

would be outlined, assuming it is a valid scrambling. As always, the challenges are designed in such a way that Peggy could meet *both* of them only if she knew how to satisfy the circuit, but answering either *one* of them yields no information about how. As previously argued, because Peggy cannot predict ahead of time which challenges will be issued by Vic, each round increases Vic’s confidence in Peggy’s claim.

The “scrambling” of each truth table by Peggy consists of a random row permutation and column complementation. Let us illustrate this principle with an example. Figure 2(a) shows the truth table for the Boolean conjunction (“AND”). The rows of this table are randomly permuted to yield the table given in figure 2(b). (Each of the 24 possible permutations — including the identity permutation — may be chosen with uniform probability.) Then, one bit is randomly chosen for each of the three columns of the truth table. Finally, each column is complemented if and only if its corresponding random bit is a 1, as shown in the three intermediate tables. The final result is illustrated in figure 2(c). Notice that the whole scrambled truth table can still be unmistakably recognized as representing the Boolean conjunction (provided the complementation bits, shown within circles throughout the drawings, are specified).

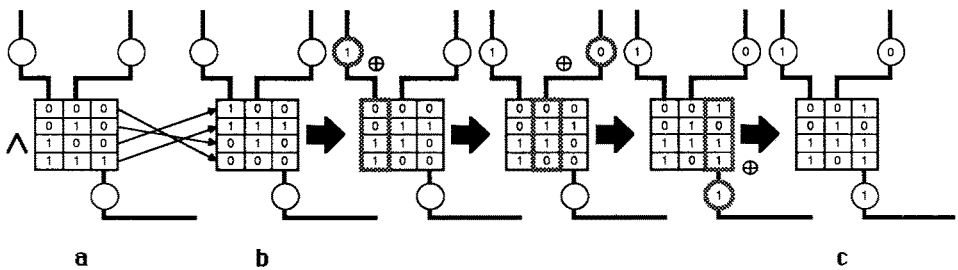


Figure 2. Permutation and complementation of a truth table.

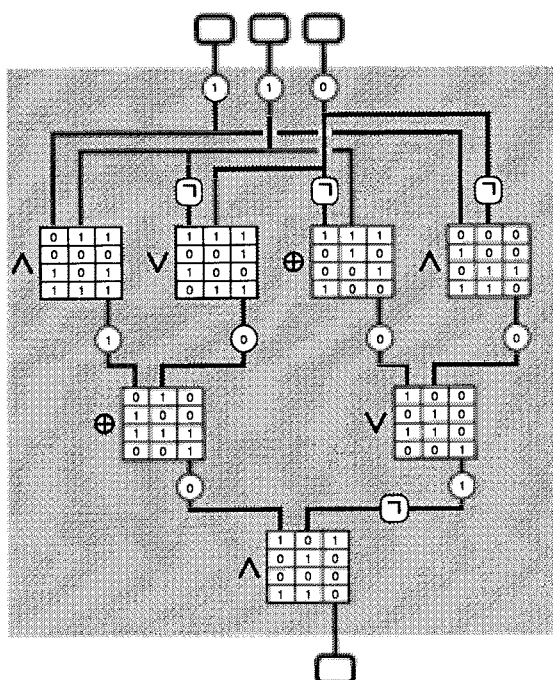


Figure 3. A circuit with randomly permuted and complemented truth tables.

The complementations must be chosen consistently: all truth table columns corresponding to the same wire in the circuit must either all be complemented or all remain the same. This is achieved by choosing randomly and independently the complementation bits corresponding to each wire. (For simplicity, we never complement the output of the final gate.) Figure 3 gives the result of some example random permutations and complementations of the truth tables in our original circuit from figure 1.

After producing a circuit similar to that of figure 3, Peggy commits to it: for each truth table bit, Peggy commits to a blob that she knows how to open accordingly. (It is not necessary for her to actually commit to the complementation bits, but they must remain secret for the moment.) One may think

of Peggy having drawn figure 3 on the floor but having covered its bits with opaque tape before allowing Vic to look. Now that the “cut” is completed by Peggy, it is time for Vic to “choose”: Vic asks Peggy to convince him of her good faith by requesting that she answers, at his random choice, either Challenge “A” or Challenge “B”, defined as follows:

- If the challenge is “A”, Peggy must open each and every blob she just committed to. Moreover, she must also reveal all the complementation bits that she used in the scrambling process. Continuing our intuitive image, Peggy strips off all the tape in order to show Vic the equivalent of figure 3. This allows Vic to verify that the information concealed by the blobs corresponds to valid permutations and complementations of the Boolean circuit’s truth tables.
- If the challenge is “B”, Peggy opens only the blobs corresponding to one row in each truth table. The rows to be opened are precisely those that were outlined in figure 1 in their (probably) new location determined by the row permutations. Still continuing our image, Peggy selectively strips off pieces of tape in order to show Vic the equivalent of figure 4. This allows Vic to verify the consistency of each wire and the fact that the final output of the circuit is a 1 bit.

Three requirements must be satisfied in order to prove the correctness of this protocol. Assuming only the four defining properties of blobs (see section 5.5), it is proved in [53] that the following holds except perhaps with an exponentially small probability:

- 1) Peggy can carry out her share of the protocol, provided she knows a satisfying assignment for Ψ . (Of course, no protocol could possibly *force* Vic to be convinced, even giving him the satisfying assignment in the clear, because he can always refuse to listen.)
- 2) If Peggy does not know a satisfying assignment for Ψ , no matter how she pretends to follow the protocol, Vic will catch her cheating.
- 3) If Peggy knows a satisfying assignment for Ψ and if she faithfully follows her share of the protocol, she does not reveal anything to Vic that could help him determine her (or any other) satisfying assignment (or even find

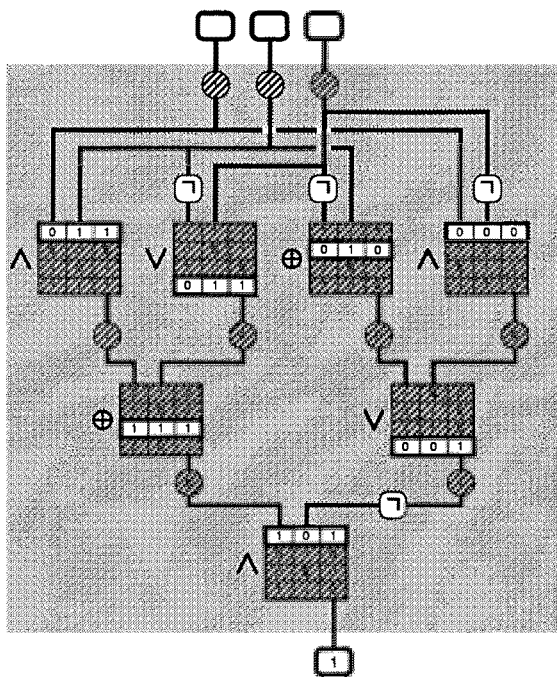


Figure 4. Showing the existence of a satisfying assignment.

partial information about it) — this remains true even if Vic deviates arbitrarily from his stipulated behaviour in the protocol.

Even though the third requirement is satisfied by our protocol, this does *not* in general imply that Vic cannot obtain anything beyond the fact that Peggy genuinely knows a satisfying assignment for Ψ . For instance, it is possible that only Peggy has the technology or knowledge necessary to commit to these blobs, in which case Vic might obtain something he could not have produced himself — although not the satisfying assignment. In other words, our protocol is minimum disclosure, but it may not be “zero-knowledge” in the terminology of [134].

Intuitively, a protocol is *zero-knowledge* if the third requirement is strengthened to the effect that Vic cannot obtain anything at all beyond learning that Peggy knows a satisfying assignment. More precisely, Vic must be able to simulate his entire conversation with Peggy without in reality ever talking to her. Refer to [134] for a formal definition. Nevertheless, our protocol is zero-knowledge provided blob defining property (iv) is strengthened (refer to section 5.5) to make sure that Vic does not gain *anything* from the process by which Peggy commits to blobs and that he obtains *only* the intended bits from the process by which Peggy opens some of them. Following the proof techniques of [134], we say that the blobs are *simulatable* if, in addition to properties (i), (ii) and (iii), they satisfy

iv') Vic can simulate what he would have been provided in the process by which Peggy commits to blobs that she could open as 0 and to blobs that she could open as 1. He can also simulate the process by which she would open these blobs had she committed to them herself.

Notice that all the blob implementations described in section 5.5 are simulatable.

If it is *information theoretically* impossible to distinguish between blobs used by Peggy to commit to 0 and blobs used to commit to 1, and if the blobs are simulatable, the interactive protocol described above for the satisfiability of Boolean expressions becomes *perfect* zero-knowledge in the terminology of [130]. This is achieved, for instance, if one uses the first blobs described in section 5.5 based on the discrete logarithm. Note that these blobs do not fit the model of [134] since an infinitely powerful prover could cheat them, which explains why Fortnow's result [113] does not apply. (Be aware that a perfect zero-knowledge interactive protocol for satisfiability was incorrectly claimed in [56].)

An interesting situation occurs if one considers a variation on the protocol in which all the rounds are carried out in parallel: Peggy commits all at once to blobs corresponding to k circuits similar to figure 3, Vic sends his string of challenges, and Peggy opens the blobs as requested by the challenges. (This would be more efficient in some settings.) Assume for simplicity that blobs are bit strings and that Peggy commits to a blob by showing it

in the clear. Consider the following strategy for Vic: after receiving from Peggy blobs corresponding to all k circuits with randomly permuted and complemented truth tables, he concatenates these blobs together and uses the result as input to some one-way function. He then uses the first k bits of the output of this function to determine the k challenges to be issued. Even though running the modified protocol with Peggy does not help Vic in learning anything about Peggy's secret, its transcript may enable him to convince someone else of the existence of this secret, because Vic could almost certainly not have produced the transcript otherwise (even if the blobs are simulatable). This leads to a curious phenomenon: the transcript of a parallel version of the protocol may contain no information on Peggy's secret (in the sense of Shannon's information theory), yet it can be used to convince someone else of the secret's existence!

If it is important that the protocol be carried out in parallel (perhaps for reasons of efficiency), it remains zero-knowledge provided defining property (iv) is strengthened further. We say that the blobs are *chameleon* if, in addition to properties (i), (ii) and (iii), they satisfy:

iv'') Vic can simulate what he would have been provided in the process by which Peggy commits to blobs. Moreover, for each of these blobs, Vic can simulate both the process by which Peggy would open it as a 0 and the process by which she would open it as a 1.

In other words, chameleon blobs allow Vic to do just what property (ii) prevents Peggy from doing. The advantage of chameleon blobs is that they allow Vic to simulate in a straightforward way his entire conversation with Peggy. This remains true even if Vic deviates arbitrarily from his stipulated behaviour. Even if Peggy and Vic have similar computing abilities, this property can sometimes be achieved if Vic has additional information. For instance, the blobs (described in section 5.5) based on graph isomorphism are chameleon if and only if Vic knows the isomorphism between the two graphs G and H concerned. Similarly, the first type of blobs based on the discrete logarithm are chameleon if and only if Vic knows the discrete logarithm of s . This is possible if, instead of choosing s randomly as specified previously, he chooses e randomly between 1 and $p-2$ and computes s as

$\alpha^e \bmod p$. However, the dual blobs based on the discrete logarithm are not chameleon, of course, since each blob determines one bit uniquely.

Is it better to trust the prover or the verifier? “Cheating” takes on a different meaning, depending on whether one is talking about Peggy or Vic. For Vic to cheat means that he learns something beyond the fact that Peggy has access to the information she claims to have. Perhaps he did not quite obtain the Hamiltonian circuit he is desperately seeking, for instance, but he learned enough to drastically reduce his search. On the other hand, for Peggy to cheat means that she succeeds in convincing Vic that she has information that would pass the certifying procedure, when in fact she does not.

It is also interesting to distinguish between *lucky* and *daring* successful cheating. The former refers to Peggy or Vic figuring out — against all odds — a piece of information that will enable him/her to quietly go about his/her cheating with the certainty of being successful and undetected. The latter refers to Peggy or Vic taking an illegal move that is almost certainly going to result in his/her cheating being detected at some point in the future, but that might nonetheless, with an exponentially small probability, allow him/her to succeed. Finally, cheating may be called *retroactive* (or *off-line*) if it can take place some time after the protocol is completed, by looking back at its transcript; it is *real-time* if it must be completed while the protocol is taking place.

If blobs unconditionally secure for Vic are used (corresponding to the protocols given by [130, 55]), Peggy could never participate with peace of mind: an algorithmic breakthrough might allow Vic to cheat retroactively, even if the new algorithm is not fast enough for a real-time response while the protocol is taking place. Even if the cryptographic assumption turns out to be well-founded, Vic still has a (very slight) probability of lucky (hence undetectable) cheating. On the other hand, regardless of any assumptions, the only cheating Peggy could attempt would be of the daring kind.

By contrast, if blobs unconditionally secure for Peggy are used (corresponding to the protocols given in [68, 56]), Vic’s belief that Peggy cannot cheat depends on his belief in the appropriate cryptographic assumption. With the first implementation of blobs based on the discrete logarithm

(described in section 5.5), for instance, Peggy could “open” any blob as either 0 or 1, whichever suits her best, if she could only obtain the discrete logarithm of s before the end of the first round in which she is asked a challenge she is not otherwise prepared to answer. (Obtaining this logarithm at any later time would be of no use to her.) Moreover, even if the cryptographic assumption is well-founded, Peggy still has a (very slight) possibility of breaking it by luck, but she must be daring to suggest conducting the protocol in the hope that she will be so lucky. With the discrete logarithm blobs, Vic has no possibility of cheating whatsoever, regardless of his luck and computing power. Finally, retroactive cheating is meaningless for either party in this context.

If blobs unconditionally secure for Peggy are used, additional security for Vic is obtained by asking Peggy to repeat the entire protocol with a different type of blob each time. In order to cheat, this would force Peggy to be capable of breaking several different cryptographic assumptions. For instance, using also the blob implementation of [56], she would need efficient on-line algorithms both for factoring and for extracting discrete logarithms. Curiously, the opposite effect is obtained with the blobs that are unconditionally secure for Vic: repeating the protocol with different types of blobs would only make it *easier* for Vic to cheat since he can do so by breaking (possibly off-line) any one of the underlying cryptographic assumptions. Nonetheless, increased security *can* be obtained if several types of blobs unconditionally secure for Vic are combined in a different way: each time Peggy wishes to commit to some bit b , she commits to one blob of each type at random except that b is the exclusive-or of the corresponding bits. Naturally, using this strategy with blobs unconditionally secure for Peggy would only make it easier for her to cheat.

Is it preferable to trust Vic or Peggy? We do not know, but it sure is nice to have the choice! Finally, consider the following provocative situation: suppose that Peggy claims to have proved Theorem T and she uses the discrete logarithm blobs to convince a skeptical Vic of this. At the end of the protocol, regardless of any unproved assumptions, Vic will be convinced that Peggy has either a proof of T or hot results on extracting discrete logarithms!

In particular, no assumptions are needed if T's claim is: "I have an efficient algorithm for the discrete logarithm problem"...

5.7. Protection of Privacy

(Written by David Chaum; printed here by kind permission)

Soon, by accessing a computerized network from almost anywhere, you may be able to pay for a purchase, order a film to be shown in your home, change your life insurance policy, or perhaps send an electronic "letter" to a friend. Although a single integrated system is of course still some way off, its piecemeal construction is already underway. Automatic cash dispensers and even direct electronic payments at shops, for instance, are already in use in many countries and planned in many more. Such systems do have enormous potential for cutting organizations' costs and increasing their security and efficiency, as well as for enhancing consumer convenience. The prevailing approach to building them, however, brings with it some very serious dangers.

This current approach requires individuals to identify themselves to the system each time they use it. All the various identifying techniques — like current plastic cards, memorized secret numbers, and fingerprints — are essentially equivalent to universal ID numbers, such as the ones used for social insurance or passports. These identifiers allow computerized linking of financial, employment, medical, and all manner of other personal data, laying bare individuals' lives to an unprecedented extent. Moreover, longstanding fears of an electronic Big Brother are being given new legitimacy by advances in "automatic pattern recognition". With these techniques, anyone tapping into the large-scale systems currently being planned could automatically categorize individuals by their transaction patterns — everywhere they pay, what organizations they have relationships with, and who they communicate with — in a form of invisible mass surveillance. Legal mechanisms, faced with the ease of tapping and modifying computerized data, seem powerless to avert such dangers or to protect individuals from the resulting potential for errors and discrimination.

All these problems can be avoided by a new approach to using personal "card computers". These would be similar to Visa-Toshiba's "super-smart"

credit card, already being tried with the current approach. It has a character display and buttons like a pocket calculator. As with all current approach cards, security relies on tamper resistance to protect against cheating by individuals to whom cards are issued. The new-approach card computer differs in that it is your own personal pocket computer: you are free to obtain its hardware and software anyway you choose, and during transactions it need never leave your hands. Instead of expensive but still imperfect tamper resistance, which provides security only to organizations, the new approach uses advanced coding techniques to enforce optimal protection for individuals and organizations alike.

Your card computer allows a transaction only after you key a secret authorizing number into it; this provision renders the card useless if stolen. You can also store backup copies of its data at home, or with friends or relatives, so that if your card is lost or damaged, you can easily initialize a new one.

Shopping with your electronic wallet.

Buying something with your card computer would be even easier than currently. The clerk enters the amount of purchase on the computerized cash register, which transmits it to your card. If you agree with the transaction details your card displays, you enter your secret authorizing number, and the cash register promptly shows “Payment Accepted”.

The system works like electronic cash: instead of paper money in various denominations, your card holds special numbers worth various amounts; the numbers themselves are legal tender. When you key in your authorization, your card computer transmits an appropriate selection of these banknote numbers to the cash register. The register checks that the note numbers correctly represent the total amount and electronically forwards them to a bank. The bank’s computer in turn verifies their validity before checking with a clearing-house to ensure that they have not been used before. The bank’s OK is then displayed by the cash register.

You get new notes for your electronic wallet by withdrawing them from a bank. Paper cash you withdraw is in principle traceable to your account,

since the serial numbers could conceivably be recorded by the bank. But with the new approach, it is as if you make a withdrawal by giving the bank a slip of paper hidden in a carbon-lined envelope, which the bank stamps with, say, a “worth-one-dollar” signature; you remove and discard the envelope before using the carbon-signed slip to make a purchase. The same principle applies in the actual electronic system, but the envelope, the slip, and the signature are specially coded numbers. You cannot counterfeit the coded signature, and the bank cannot find out which numeric “slip” was in which “envelope” it signed. In fact, your payments are “unconditionally untraceable”: a simple mathematical proof shows that they cannot be traced to your account, no matter how extensive or ingenious the computerized attack.

Showing credentials without identification.

Business and government organizations sometimes do legitimately need to see statements about individuals that are issued by other organizations. Such credentials have in the past taken the form of paper identifying certificates like passports, driver’s licenses, and membership cards. The original purpose of most such documents was to securely authenticate individuals’ qualifications, such as an economic or age bracket, a license, or an academic degree; identification was a means to that end. But in today’s computerized imitation of these paper-based methods, identification allows organizations to match with or directly access other organizations’ computerized files, in effect creating a single huge database on individuals.

Possible future extensions of the card-computer payment system prevent these problems — by getting rid of identification. What is more, they actually protect organizations’ legitimate interests better than linking and dossier-building.

Credentials are issued by organizations as coded, unforgeable signatures that your card computer handles much as it does electronic cash. Other organizations’ computers specify their requirements as various possible qualifying combinations of such credentials. If you decide to show that you qualify, your card computer can transmit convincing proof that it holds one of these combinations — but without revealing anything about which one. The coding used ensures organizations that you cannot modify, lend, or escape

accountability for your credentials. And because of the way your card codes “blank” credentials in “envelopes” before they are signed by organizations, your credentials cannot be linked any more than your payments can. You retain complete control over your personal information, just as if the computerized records that organizations maintain on you today were stored only in your card computer.

Protecting electronic mail.

Electronic mail may someday be widely used by consumers. The potential problems range from annoyances, like the sale of electronic mailing lists for computerized junk mail, to more sinister possibilities, like automated “traffic analysis” that traces and exposes the patterns of people’s relationships even when message content is securely coded.

The new approach takes care of these problems partly by broadcasting messages to all electronic mail computers on the system, which both prevents tracing to any particular recipient and ensures that the intended recipient cannot deny having received a message. Yet novel coding allows each person’s mail computer to conceal untraceably, from everyone, which messages it sends. Other coding keeps message content confidential and enables recipients to be sure — and to convince others if necessary — of the message originator’s pseudonym. This system naturally allows payment and credential transactions from home, with all the convenience and protection offered by card computers.

The moment of decision.

Organizations already have compelling reasons to adopt the new approach, including the direct incentives of increased efficiency, improved security, and reduced overall cost. As more people become aware both of the dangers posed by the current approach to automating and of the advantages offered by the alternative, these incentives may be augmented by consumer preference, public opinion, and even legislation. Moreover, once it becomes apparent that there is no good reason not to adopt the new approach, an organization that rejects it may be suspected of wanting to gather information so as to clandestinely monitor and control individuals.

Currently, however, the centralized-data approach is gaining institutional momentum. The use of automatic cash dispensers and direct electronic payment, with the tracing they allow, are just the tip of the massive investment iceberg required by these large-scale systems.

We are fast approaching a moment of crucial and perhaps irreversible decision, not merely between two kinds of transaction system, but between two kinds of society. If the current trend continues, the enormous surveillance potential of such automated systems will not only render hollow most legal safeguards on privacy and other protections, but may seriously chill participation in public and political life. If, on the other hand, the new approach prevails, erosion of individuals' informational rights can be reversed and new rights added — notably the right, made possible by personal card computers, to reveal only necessary information in transactions. In an age of pervasive computerization, control over information is the key to social, economic, and political power. Card computers would restore a balance by putting part of that key, both literally and figuratively, back in the hands of ordinary citizens.

For more information on the topics discussed in this section, read [65, 67, 66, 69].

5.8. Additional Applications

Cryptographic techniques have so many diverse applications that we shall not even attempt to describe them all here. Some of these applications are still mostly academic; others are already quite practical. Let us briefly mention some of the most significant achievements and refer you to the literature for more information.

Oblivious transfer allows one party to send a message to another party in such a way that the message has exactly a 50% probability of being received. This probability cannot be influenced by either party. This notion is due to Rabin [198]. For applications, consult [34, 198]. Read also [27, 84].

A fun variation on coin flipping is playing games in which randomness plays a role. Of particular interest is *mental poker*. How can two or more parties play poker over the telephone if none of them trust any of the others?

A first proposal (together with a proof of impossibility against unlimited computing power in the two-player case) was given by Shamir, Rivest and Adleman [213], but it was shown insecure by Lipton and Coppersmith because it could leak partial information on the cards that are supposed to be hidden [171, 78]. This particular difficulty was solved by Goldwasser and Micali [132] as an application of probabilistic encryption, and a full solution was given, but there was no obvious way to extend this protocol to more than two players. An unconditionally secure solution was given by Banary and Furedi [10] as long as there are at least three players and coalitions are not allowed. More and more subtle solutions were proposed by various people [248, 114, 82]. The ultimate (theoretical) solution was finally given by Crépeau [83], which even allows the players to keep an electronic poker face! For a generalization to any mental game, consult Goldreich, Micali and Wigderson [131].

You own a valuable secret, such as the factorization of your master key. If you keep it locked in your office, you run the risk of loosing it in the advent of a fire; if you keep several copies in various places, you increase the risk of one of them falling into someone else's hands. An (n, k) -*threshold scheme* allows you to *share your secret* into n *shadows* in a way that it suffices for you to access any k of them to recover the secret, whereas any $k - 1$ shadows yield *no* information on the secret. This notion was developed independently by Shamir and Blakley [208, 30, 164].

Each of two parties knows a different secret, perhaps the factorization of their respective RSA moduli. Suppose they are willing to exchange them. How can this *exchange of secrets* be performed without either party running the risk of finding out at the end of the protocol that he has just been given junk in exchange for a genuine secret of his? This question was first investigated by Blum, who was subsequently followed by many researchers [34, 198, 36, 42, 173, 224, 225, 144, 245].

Certified mail is a sophisticated version of digital signature. In this problem, one party wishes to send a message to another party in a way that the second party gets the message if and only if the first one obtains a signed receipt to this effect. Blum and Rabin have worked on this problem [34, 36, 199, 42, 231]. The protocol implements *all-or-nothing* certified mail, a

notion also introduced by Blum, if the receiver learns *absolutely* nothing about the contents of the message until the sender has his receipt, and *vice versa* [37].

Contract signing is even more sophisticated (although it is easy to implement given certified mail as a primitive). It allows two parties to sign a contract over a computer network in a way that neither party can stop the protocol early (except with small probability) and dispose of the other's signature while he is not yet committed to the contract [36, 199, 125, 42, 107].

Assume you dispose of some number of secrets and you are willing to sell one of them. A *disclosure protocol* allows you to do so in a way that the buyer gets to choose which secret he obtains and you don't get to know which it was. The protocol is *all-or-nothing* if the buyer is prevented from obtaining bits and pieces from several secrets. This problem was addressed and solved by Brassard, Crépeau and Robert [57, 58] (and also independently by Chaum). This is useful in particular to implement a multi-party mental poker protocol such as Crépeau's [83], and it can serve as primitive for more sophisticated cryptoprotocols.

Assume you share a truly random bit string with a friend, but you suspect some of it might have fallen into the hands of an eavesdropper. Given only an upper bound on the *amount* of information compromised, it is possible to *amplify your privacy by public discussions* so as to end up with a shorter but much more secret bit string. This remains possible even if transmission errors and malicious tampering could have caused your friend's original string to be slightly different from yours. This question was investigated independently by Bennett, Brassard and Robert [24] and by Chor, Goldreich, Hastad, Friedman, Rudich and Smolensky [75]. It is a crucial building block for the establishment of the quantum channel, as described in the next chapter.

A *poly-random collection of functions* is a set of *efficiently* computable functions that are polynomially indistinguishable from truly random functions. This notion was invented by Goldreich, Goldwasser and Micali [129], who also gave numerous applications for them [128], such as computationally secure authentication tags requiring short secret shared keys. Poly-random collections of *permutations* were constructed by Luby and Rackoff [174].

Strangely enough, these permutations are obtained through a DES-like process.

A *subliminal channel* allows two levels of encryption. The *secret message* can be recovered from the cryptogram using the “regular” secret key, but an additional key is needed to obtain the *subliminal message*. This notion, invented by Simmons, allows two parties to communicate secretly while an unsuspecting eavesdropper is fooled into thinking he can read their communications [219, 220].

Although no truly practical cryptographic scheme for *software protection* has yet been proposed, interesting theoretical ideas have been proposed by Herzberg and Pinter [149] and by Goldreich [127]. In particular, Goldreich distinguishes between protection against illegal copying and illegal distribution [127].

Yao’s *millionaire’s problem* goes as follows: two millionaires wish to know which is the richest in a way that neither gets to know the other one’s wealth [244]. This is a special case of *secure distributed computing*, by which two parties wish to compute the value of some agreed upon function $f(a, b)$ on inputs a secretly known by one party and b secretly known by the other, in a way that neither party gets to learn any information on the other’s secret parameter (except for what can be inferred from the value of $f(a, b)$ and one’s own secret). As an application, Yao showed that it allows two parties to collaborate at choosing some integer that they know to be the product of exactly two primes, in a way that neither party can figure out its factorization unless they cooperate further [245].

Secure distributed computing generalizes naturally to *any number of parties*. Galil, Haber and Yung have studied secure fault-tolerant protocols in the public-key model [118]. General cryptographic solutions to the problem of secure distributed computing have been given by Goldreich, Micali and Wigderson [131] and Chaum, Damgård and van de Graaf [71, 85]. It was later shown independently by Chaum, Crépeau and Damgård [70] and Ben-Or, Goldwasser and Wigderson [25] that unconditionally secure distributed computing can be achieved under the only assumption that no more than one-third of the participants are dishonest.

An *election scheme* is a special case of the previous problem. How can several parties cast their votes over a computer network in a way that they can figure out the outcome of the vote but keep secret all individual votes? This question has been studied by many researchers, including Cohen (Benaloh) and Fisher [76], Ben-Or and Linial [26] and Benaloh (Cohen) and Yung [15].

Another interesting problem is that of *computing with encrypted data*, an idea first investigated by Feigenbaum [110] and refined subsequently by Abadi, Feigenbaum and Kilian [1]. Here, one party wishes to know the value of $f(x)$ for some public function f and secret data x , but he lacks the power (or knowledge) to compute it. The other party has this capability and is willing to help the first one. The functions f is *encryptable* if the first party can use the second one to help him compute $f(x)$ in a way that does not reveal *anything* about x , not even the value of $f(x)$. Brassard and Bratley pointed out a direct correspondence between this notion and the algorithmic design technique known as *stochastic preconditioning* [52].

For other marvels, you can also read [2, 32, 218, 172], and browse through the proceedings of the CRYPTO, EUROCRYPT, FOCS and STOC conferences and through the *Journal of Cryptology* and *Cryptologia*. In addition, other scientific journals regularly produce special issues devoted to cryptology, such as the May 1988 issue of the *Proceedings of the IEEE*.

QUANTUM CRYPTOGRAPHY

(Chapter written with Charles H. Bennett)

6.1. Introduction

When elementary quantum systems, such as polarized photons, are used to transmit digital information, the uncertainty principle of quantum physics [43] gives rise to novel cryptographic phenomena, unachievable with ordinary transmission media. For instance, one can obtain a communication channel on which it is impossible in principle to eavesdrop without a high probability of disturbing the transmission in such a way as to be detected. Such a channel achieves one of the main advantages of public key cryptography: it permits secure distribution of key information between two parties who share no secret information initially, provided the parties have access, besides the quantum channel, to an ordinary channel susceptible to passive eavesdropping but not active tampering. (Even in the presence of active tampering, the two parties can still distribute a key securely if they share some secret information initially, provided that the tampering is not so active as to suppress communication completely.) The resulting *quantum channel* is provably secure even against an opponent with superior technology and unlimited computing power (even if $P=NP!$), barring fundamental violation of accepted physical laws.

In conventional information theory and cryptography, it is taken for granted that digital communications can always be passively monitored or copied, even by someone ignorant of their meaning. This can be useful for instance if one hopes to be able to figure out (or spy out) the key some time in the future, perhaps after a sufficient volume of ciphertext has been obtained. By contrast, when information is encoded in non-orthogonal quantum states, such as single photons with polarization directions 0, 45, 90 and

135 degrees, one obtains a communication channel whose transmissions in principle cannot be read or copied reliably by an eavesdropper ignorant of certain key information used in forming the transmission. The eavesdropper cannot even gain partial information about such a transmission without altering it in a random and uncontrollable way that is likely to be detected by the channel's legitimate users.

A disadvantage of this approach in practice is the fact that quantum transmissions are necessarily very weak and cannot be amplified in transit. Moreover, quantum cryptography cannot provide digital signature (section 5.2) and related features such as certified mail (section 5.8) or the ability to settle disputes before a judge. (However, these limitations can be argued to be inherent to any scheme secure against unlimited computing power.) Nevertheless, Brickell and Odlyzko close their very thorough survey of recent results in cryptanalysis by these words: “If such systems [quantum cryptography] become feasible, the cryptanalytic tools discussed here [in their paper] will be of no use.” [62].

Quantum coding was first described by Wiesner [237], along with two applications: making money that is in principle impossible to counterfeit, and multiplexing two or three messages in a way that reading one destroys the other(s) (much like the all-or-nothing disclosure protocol described in section 5.8 [57, 58]). More than a decade later, Bennett, Brassard, Breidbart and Wiesner [23] showed how to use quantum coding in conjunction with techniques from public key cryptography to yield several schemes for unforgeable subway tokens. Later, Bennett and Brassard [17, 18] devised the quantum channel described above, as well as *quantum coin flipping* (see section 5.4 for a discussion of “coin flipping”). This quantum coin flipping protocol yields naturally the notion of *quantum blobs* for bit commitment (section 5.5) [22]. In this chapter, only the basic quantum channel is described, together with its use for public key distribution. A complete discussion of quantum cryptography will be provided in the forthcoming paper [22].

6.2. Essential Properties of Polarized Photons

Polarized light can be produced by sending an ordinary light beam through a polarizing apparatus such as a Polaroid filter or calcite crystal. The beam's polarization axis is determined by the orientation of the polarization apparatus in which the beam originates. Generating single polarized photons is possible in principle by picking them out of a polarized beam, but it may be technologically infeasible to do so. The next section assumes for simplicity the availability of single photons with definite polarizations, but this assumption is removed in section 6.4.

Although polarization is a continuous variable, the uncertainty principle forbids measurements on any *single* photon from revealing more than one bit (in a probabilistic sense) about its polarization. For example, if a light beam with polarization axis α is sent into a filter oriented at angle β , the individual photons behave dichotomously and probabilistically, being transmitted with probability $\cos^2(\alpha - \beta)$ and absorbed with the complementary probability $\sin^2(\alpha - \beta)$. The photons behave deterministically only when the two axes are either parallel (certain transmission) or perpendicular (certain absorption). (Any other elementary two-state quantum system, such as a spin $\frac{1}{2}$ atom, behaves similarly dichotomously and probabilistically.)

If the two axes are not perpendicular, so that some photons are transmitted, one might hope to learn additional information about α by measuring the transmitted photons again with a polarizer oriented at some third angle. However, this is of no avail because the transmitted photons, in passing through the β polarizer, emerge with exactly β polarization, having lost all “memory” of their previous polarization α . Of course, if it is known that a beam is composed of several photons of identical polarization, different measurements can be made on the distinct photons, so as to get more than one bit of information about the common polarization.

Another way one could hope to learn more than one bit from a single photon would be not to measure it directly, but rather somehow amplify it into a clone of identically polarized photons in order to subsequently perform different measurements on these. However, this hope is also vain because

such cloning can be shown to be inconsistent with the foundations of quantum mechanics [241].

Formally, quantum mechanics represents the internal state of a quantum system (such as the polarization of a photon) as a vector of unit length in a linear space over the field of complex numbers (Hilbert space). The dimensionality of the Hilbert space depends on the system, being larger (or even infinite) for more complicated systems. Each physical measurement that might be performed on the system corresponds to a resolution of its Hilbert space into orthogonal subspaces, one for each possible outcome of the measurement. The number of possible outcomes is thus limited to the dimensionality d of the Hilbert space. The most complete measurements are those that resolve the Hilbert space into d 1-dimensional subspaces.

The Hilbert space for a single polarized photon is 2-dimensional. Thus, the state of a photon may be completely described as a linear combination of, for instance, the two unit vectors $r_1 = (1, 0)$ and $r_2 = (0, 1)$, representing respectively horizontal and vertical polarizations. In particular, a photon polarized at angle α to the horizontal is described by the state vector $(\cos \alpha, \sin \alpha)$. When subjected to a measurement of horizontal versus vertical polarization, such a photon in effect chooses to become horizontal with probability $\cos^2 \alpha$ and vertical with probability $\sin^2 \alpha$. The two orthogonal vectors r_1 and r_2 thus exemplify the resolution of a 2-dimensional Hilbert space into 2 orthogonal 1-dimensional subspaces. Henceforth, r_1 and r_2 will be said to comprise the *rectilinear basis* for the Hilbert space.

An alternative basis for the same Hilbert space is provided by the two *diagonal* basis vectors $d_1 = (\gamma, \gamma)$ and $d_2 = (\gamma, -\gamma)$, where γ stands for half the square root of 2. In this basis, d_1 represents a 45-degree photon and d_2 represents a 135-degree photon. Wiesner defines two bases (such as rectilinear and diagonal) to be *conjugate* if each vector of one basis has equal-length projections onto all vectors of the other basis. In simple terms, this means that a system prepared in a specific state of one basis will behave entirely randomly, and lose all its stored information, when subjected to a measurement corresponding to the other basis. Owing to the complex nature of its coefficients, the two-dimensional Hilbert space also admits a third basis

conjugate to both the rectilinear and diagonal bases, comprising the two so-called *circular* polarizations $c_1 = (\gamma, \gamma i)$ and $c_2 = (\gamma i, \gamma)$, where i stands for the square root of -1 . However, the rectilinear and diagonal bases are all that will be needed for the establishment of quantum public key distribution.

From a practical point of view, it suffices to understand that there is a simple apparatus that can distinguish between horizontally and vertically polarized photons. Another apparatus can distinguish between the two diagonal polarizations. If the first apparatus is used on a diagonal photon (or the second on a rectilinear photon), however, it behaves in a completely random and unpredictable fashion. In practice, these devices are not perfect, but they can be built in such a way that they almost never give a false answer, although sometimes they do not give an answer at all. For instance, an apparatus designed to distinguish between the two types of rectilinear photons could answer: “the photon was vertical”, “the photon was horizontal” or “I cannot tell how the photon was polarized”. If the incoming photon was indeed rectilinearly polarized, either of the first two answers would almost certainly be correct, and the third answer is not given overwhelmingly often.

6.3. Quantum Public Key Distribution

The purpose of quantum public key distribution is to use the quantum channel to transmit a supply of random bits between two users who share no secret information initially. This is achieved in such a way that the users, by subsequent conversation over an ordinary non-quantum channel (subject to unlimited eavesdropping) can tell with high probability whether the original quantum transmission has been disturbed in transit as it would be by an eavesdropper. (It is the quantum channel’s peculiar virtue to compel any type of eavesdropping to in fact result in active tampering.)

If the quantum transmission has not been disturbed, the users can safely apply these shared secret bits as secret key in a traditional cryptosystem (such as the one-time pad — section 3.2) to conceal the meaning of subsequent communications. Alternatively, these bits can be used for other cryptographic purposes requiring shared secret random information, such as Wegman-Carter authentication tags based on universal hashing [236] (section 5.1). On the

other hand, if the transmission is found to have been disturbed, the users discard the bits just obtained, and they try again. Thus, they defer any meaningful communication until they have succeeded in transmitting enough random bits through the quantum channel to serve their intended purpose. Although the eavesdropper can prevent communication between the users by jamming the channel, he cannot fool them into thinking they have succeeded when in fact they have not.

Let us now examine in more detail how two users, Alice and Bob, can perform public key distribution on the quantum channel. We assume the presence of an eavesdropper, called Eve. As a first step, Alice chooses a random bit string and a random sequence of polarization bases (rectilinear or diagonal). She then sends Bob a train of photons, each representing one bit of the string in the basis chosen for that bit position (for instance, a horizontal or 45 degree photon can be used to represent a binary 0, whereas a vertical or 135 degree photon can stand for a binary 1).

The good news is that if Eve wants to measure (some of) the photons that Alice sends to Bob, she does not know in which bases to measure them. The apparent bad news is that Bob does not know either which bases to use. Nevertheless, as Bob receives the photons, he decides, randomly for each photon and independently of Alice, whether to measure the photon's rectilinear or diagonal polarization. He then interprets each result as a binary 0 or 1, depending on the outcome of the corresponding measurement. As explained in section 6.2, a random answer is produced and all information is lost when an attempt is made to measure the rectilinear polarization of a diagonal photon or *vice versa*. Thus Bob obtains meaningful data from only about half the photons he detects (those for which he guesses the correct polarization basis), and he does not yet know which ones these are. Bob's information is further degraded by the fact that, in practice, some of the photons will be lost in transit or would fail to be counted by Bob's imperfect detectors.

Subsequent steps of the protocol take place over an ordinary public communication channel. Let us first assume that this channel is susceptible to eavesdropping but not to the injection or alteration of messages. Bob and Alice first determine, by public exchange of messages, which photons were

successfully received and of these which were measured by Bob in the correct basis. If the quantum transmission has been undisturbed, Alice and Bob should now agree on the bits encoded by these photons, even though this data has never been discussed over the public channel. Each of these photons, in other words, presumably carries one bit of random information (such as whether a rectilinear photon was vertical or horizontal) known to Alice and Bob but to no one else.

Because of the random mix of rectilinear and diagonal photons in the quantum transmission, any eavesdropping carries the risk of altering the transmission in such a way as to produce disagreement between Alice and Bob on some of the bits on which they think they should agree. Specifically, it can be shown that no measurement on a photon in transit, by an eavesdropper who is informed of the photon's original basis only after he has performed his measurement, can yield more than $\frac{1}{2}$ expected bit of information about the key bit encoded by that photon. Moreover, any such measurement yielding b bit of expected information ($b \leq \frac{1}{2}$) must induce a disagreement with probability at least $b/2$ if the measured photon (or any attempted forgery of it) is subsequently re-measured (by Bob) in its original basis. (This optimum tradeoff for Eve occurs, for example, when she measures and retransmits all intercepted photons in the rectilinear basis. In this case, she learns the correct polarizations of half the photons and induces disagreements in $\frac{1}{4}$ of those that are later re-measured by Bob in the original basis.)

It remains to be seen how Alice and Bob can decide whether their resulting bit strings are identical (indicating with high probability that no eavesdropping has occurred on the quantum channel, or else that eavesdropping has been on very few photons), or whether they are different (indicating that the quantum channel has been subjected to serious eavesdropping). A simple solution is for Alice and Bob to publicly compare some of the bits on which they think they should agree. The position of these “quality control” bits must be chosen randomly after the quantum transmission has been completed, so as to deprive Eve of information about which photons she can measure safely. Of course, this process sacrifices the secrecy of these bits. If the bit positions used in this comparison are a random subset (say one third)

of the correctly received bits, eavesdropping on more than a few photons is unlikely to escape detection. If all the comparisons agree, Alice and Bob can conclude that the quantum transmission has been free of significant eavesdropping. Therefore, most of the remaining bits that were sent and received with the same basis can safely be used as a one-time pad for subsequent communication over the public channel. When this one-time pad is used up, the protocol is repeated to send a new body of random information over the quantum channel. The protocol we have just described is illustrated in figure 5.

This eavesdropping-detection protocol is rather wasteful because a significant proportion of the bits must be sacrificed to obtain a good probability that eavesdropping is detected even if attempted by Eve on only a few photons. Moreover, the probability that the resulting strings of Alice and Bob agree completely cannot be made overwhelmingly close to 1, unless most of the originally transmitted bits are sacrificed. Both these difficulties can be resolved by a more subtle verification protocol based on universal hashing [63], due to Bennett, Brassard and Robert [24], which we do not describe here. The same paper gives an efficient way for Alice and Bob to *amplify their privacy* (section 5.8), by reducing the amount of information known by Eve on their final string.

The need for the public (non-quantum) channel in this scheme to be immune from active tampering can be relaxed if Alice and Bob have agreed beforehand on a small secret key, which they use to create Wegman-Carter authentication tags [236] for their messages over the classical channel. In more detail, the Wegman-Carter multiple-message authentication scheme uses a short random key to produce a message-dependent tag for an arbitrary long message. This is done in such a way that Eve, who is ignorant of this short key, has only a negligible probability of being able to generate any other valid message-tag pair (this would be true even if Eve had unlimited computing power). The tag thus provides evidence that the message is legitimate, and was not generated or altered by Eve. These key bits are gradually used up in the Wegman-Carter scheme, and they cannot be reused without compromising the system's provable unconditional security. However, in the

1.	0	1	1	0	1	1	0	0	1	0	1	1	0	0	1
2.	D	R	D	R	R	R	R	R	D	D	R	D	D	D	R
3.															
4.	R	D	D	R	R	D	D	R	D	R	D	D	D	D	R
5.	1		1		1	0	0	0		1	1	1		0	1
6.	R		D		R	D	D	R		R	D	D		D	R
7.			OK		OK			OK				OK		OK	OK
8.			1		1			0				1		0	1
9.					1									0	
10.					OK									OK	
11.			1					0				1			1

Quantum Transmission

1. Alice's random bits
2. Random sending bases chosen by Alice
3. Photons sent by Alice
4. Random receiving bases chosen by Bob
5. Bits as received by Bob

Public Discussion

6. Bob reports the bases in which he measured the received photons
7. Alice says which bases were correctly guessed
8. This information is presumably shared (if no eavesdropping occurred)
9. Bob reveals some key bits chosen at random
10. Alice confirms these bits

Outcome

11. Remaining shared secret bits

Figure 5. Quantum public key distribution.

present application, these key bits can be replaced by fresh random bits successfully transmitted through the quantum channel. Eve can still prevent communication between Alice and Bob by suppressing messages in the public channel, as of course she can by suppressing or excessively perturbing the photons sent through the quantum channel. Once again, however, Alice and Bob would then with high probability conclude that their communications are being suppressed, and they will not be fooled into thinking otherwise.

6.4. Practical Considerations

The quantum public distribution scheme described in the previous section is very nice theoretically, but it is also impractical with current technology. In particular, the following two problems (among others) were ignored.

- 1) It is much easier to deal with beams (or pulses) of photons than with single photons.
- 2) Even if no eavesdropping is attempted on the quantum channel, and even when Bob guesses Alice's chosen basis correctly, it is to be expected that some of the photons will be repolarized in transit or that errors in the measuring apparatus will cause Bob to obtain an incorrect reading.

Nevertheless, both these difficulties can be resolved in practice. A detailed description of the proposed quantum cryptography apparatus and of the corresponding protocols is given in a technical disclosure written by Bennett and Brassard [20]. Only the most basic ideas are mentioned below.

The first problem is solved by sending very faint pulses of light whenever the ideal protocol of section 6.3 would send single photons. Using a laser, it is easy to produce such pulses in which the number of photons follows a Poisson distribution of known mean and variance. Moreover, the use of polarizers, beam-splitters and neutral-density filters allows to produce these pulses in a definite polarization, and the desired polarization can be switched very rapidly from one pulse to the next (there are no moving parts in the design).

The problem with pulses (compared to single photons) is that each time that more than one photon is issued in the same time slot, Eve gets a chance (at least in principle) to measure one of these photons in a randomly chosen basis. Such eavesdropping would be undetectable provided Eve is careful to let the other photon(s) of the same pulse continue undisturbed towards Bob. The main idea to greatly reduce this threat is to use *very* faint pulses indeed. If the expected number of photons in each pulse is one thousandth (10^{-3}), for instance, one would expect multi-photon pulses to be produced only about once every two million pulses. Therefore, 99.95% of the non-empty pulses contain a single photon. A very powerful Eve could detect the rare occurrences in which more than one photon is transmitted and learn the

corresponding bit with probability one-half by measuring one of the photons in a randomly chosen basis, but this would only result in Eve knowing about 0.025% of the bit string exchanged between Alice and Bob. (An even more powerful Eve could keep this additional photon without measuring it, until the public discussion between Alice and Bob reveals in which basis this photon should be read – thus increasing her rate of success to 0.05%.) The techniques of privacy amplification already mentioned [24] can subsequently be used by Alice and Bob to reduce below any fraction of one bit the amount of information known by Eve on their final string.

The second problem with the naive implementation of quantum cryptography described in section 6.3 is that some bits can be received incorrectly even in the absence of eavesdropping, due to imperfections in the apparatus. Alice and Bob might never succeed in exchanging a random string if they insist in starting over each time they detect even a single error in the quantum transmission. Therefore, a public channel protocol is needed to detect and correct a reasonable number of differences between Alice's original random string and Bob's received string. A suitable way of correcting these errors is for Alice to use an agreed upon error-correcting code, such as a non-systematic convolutional code, to generate a check sequence. This sequence can then be sent to Bob over the public channel. If the code has sufficient redundancy, Bob can uniquely decode the information available to him in order to recover Alice's string with high probability. Once again, the information thus given away to Eve can subsequently be nearly wiped out by use of privacy amplification protocols [24].

The amount of required redundancy depends on the expected probability that bits are received incorrectly when no eavesdropping occurs. Since Alice and Bob are in a two-way communication, they can arrive at the correct redundancy even if they do not know this probability beforehand: Alice computes a relatively large number of check bits, but sends only as many as Bob finds he needs for efficient decoding. When Bob is satisfied that all errors are corrected, and that with high probability he knows Alice's string exactly, he tells Alice the actual proportion of errors he obtained through his measurements. From this proportion of errors, the original pulse intensity and the

quantum efficiency of Bob's detectors, it is possible to estimate the extent of eavesdropping. If it is found that eavesdropping probably occurred but that Eve is likely to have obtained only a small amount of information, it is possible to continue safely the key distribution, still using the privacy amplification protocols to deprive Eve from nearly all the information she obtained. On the other hand, when an excessive amount of eavesdropping is detected, Alice and Bob can no longer rule out the possibility that Eve may have nearly complete information. In this case, they must discard their data and begin over again using a new quantum transmission.

A real implementation has to take account of several additional constraints. A more technical explanation of how this could be achieved is given in detail in [20].

BIBLIOGRAPHY

1. ABADI, M., FEIGENBAUM, J. and KILIAN, J., "On hiding information from an oracle", *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987, pp. 195–203; final version submitted to *Journal of Computer and System Sciences*.
2. ADLEMAN, L. M., "Implementing an electronic notary public", *Advances in Cryptology: Proceedings of CRYPTO 82*, Plenum Press, 1983, pp. 259–265.
3. ADLEMAN, L. M., "On breaking generalized knapsack public key cryptosystems", *Proceedings of the 15th ACM Symposium on Theory of Computing*, 1983, pp. 402–412.
4. AHO, A.V., HOPCROFT, J.E. and ULLMAN, J.D., *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.
5. ALEXI, W., CHOR, B.-Z., GOLDREICH, O. and SCHNORR, C.P., "RSA and Rabin functions: Certain parts are as hard as the whole", *SIAM Journal on Computing*, to appear; extended abstract in *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, 1984, pp. 449–457.
6. ALPERN, B. and SCHNEIDER, B., "Key exchange using 'keyless cryptography'", *Information Processing Letters*, vol. 16, 1983, pp. 79–81.
7. ANGLUIN, D. and LICHTENSTEIN, D., "Provable security of cryptosystems: A survey", Technical Report YALEU/DCS/TR-288, Department of Computer Science, Yale University, 1983.
8. BABAI, L., "Trading group theory for randomness", *Proceedings of the 17th ACM Symposium on Theory of Computing*, 1985, pp. 421–429.
9. BACON, R., *Secret Works of Art and the Nullity of Magic, circa 1250*.
10. BANARY, I. and FUREDI, Z., "Mental poker with three or more players", *Information and Control*, vol. 59, 1983, pp. 84–93.
11. BANERJEE, S. K., "High speed implementation of DES", *Computers and Security*, vol. 1, 1982, pp. 261–267.
12. BEAUCHEMIN, P. and BRASSARD, G., "A generalization of Hellman's extension to Shannon's approach to cryptography", *Journal of Cryptology*, vol. 1, 1988, to appear.
13. BEAUCHEMIN, P., BRASSARD, G., CRÉPEAU, C., GOUTIER, C. and POMERANCE, C., "The generation of random numbers that are probably prime", *Journal of Cryptology*, vol. 1, 1988, to appear.
14. BENALOH (COHEN), J.D., "Cryptographic capsules: A disjunctive primitive for interactive protocols", *Advances in Cryptology — CRYPTO '86 Proceedings*, Springer-Verlag, 1987, pp. 213–222.

15. BENALOH (COHEN), J. D. and YUNG, M., "Distributing the power of a government to enhance the privacy of voters", *Proceedings of the 5th ACM Symposium on Principles of Distributed Computing*, 1986.
16. BENGIO, S., BRASSARD, G., DESMEDT, Y., GOUTIER, C. and QUISQUATER, J.-J., "Aspects and importance of secure implementations of identification systems", Manuscript M209, Philips Research Laboratory, Bruxelles, Belgium, 1987.
17. BENNETT, C. H. and BRASSARD, G., "Quantum cryptography, and its application to provably secure key expansion, public-key distribution, and coin tossing", *Abstracts of Papers from IEEE International Symposium on Information Theory*, St-Jovite, Québec, 1983, p. 91.
18. BENNETT, C. H. and BRASSARD, G., "Quantum cryptography: Public-key distribution and coin tossing", *Proceedings of the International Conference on Computers, Systems and Signal Processing*, Bangalore, India, 1984, pp. 175—179.
19. BENNETT, C. H. and BRASSARD, G., "An update on quantum cryptography", *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 475—480.
20. BENNETT, C. H. and BRASSARD, G., "Quantum public key distribution system", *IBM Technical Disclosure Bulletin*, vol. 28, 1985, pp. 3153—3163.
21. BENNETT, C. H. and BRASSARD, G., "Quantum public key distribution reinvented", *SIGACT News*, vol. 18, no. 4, 1987, pp. 51—53.
22. BENNETT, C. H. and BRASSARD, G., "Quantum cryptography", in preparation.
23. BENNETT, C. H., BRASSARD, G., BREIDBART, S. and WIESNER, S., "Quantum cryptography, or unforgeable subway tokens", *Advances in Cryptology: Proceedings of CRYPTO 82*, Plenum Press, 1983, pp. 267—275.
24. BENNETT, C. H., BRASSARD, G. and ROBERT, J.-M., "Privacy amplification by public discussion", *SIAM Journal on Computing*, vol. 17, 1988, to appear.
25. BEN-OR, M., GOLDWASSER, S. and WIGDERSON, A., "Completeness theorems for non-cryptographic fault-tolerant distributed computation", *Proceedings of the 20th ACM Symposium on Theory of Computing*, 1988, to appear.
26. BEN-OR, M. and LINIAL, N., "Collective coin flipping, robust voting schemes and minima of Banzhaf values", *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, 1985, pp. 408—416.
27. BERGER, R., PERALTA, R. and TEDRICK, T., "A provably secure oblivious transfer protocol", *Advances in Cryptology: Proceedings of EUROCRYPT 84*, Springer-Verlag, 1985, pp. 379—386.
28. BERLEKAMP, E. R., *Algebraic Coding Theory*, McGraw-Hill, New York, NY, 1968; revised edition: Aegean Park Press, Laguna Hills, CA, 1984.

29. BLAKE, I. F., FUJI-HARA, R., MULLINS, R. C. and VANSTONE, S. A., "Computing logarithms in finite fields of characteristic two", *SIAM Journal on Algebraic Discrete Methods*, vol. 5, 1984, pp. 276–285.
30. BLAKLEY, G. R., "Safeguarding cryptographic keys", *Proceedings of AFIPS National Computer Conference*, vol. 48, 1979, pp. 313–317.
31. BLAKLEY, G. R. and BOROSH, I., "Rivest-Shamir-Adleman public key cryptosystems do not always conceal messages", *Computers & Mathematics with Applications*, vol. 5, 1979, pp. 169–178.
32. BLAKLEY, G. R. and MEADOWS, C., "A database encryption scheme which allows the computation of statistics using encrypted data", *Proceedings of IEEE Symposium on Computer Security and Privacy*, 1985, pp. 116–122.
33. BLUM, L., BLUM, M. and SHUB, M., "A simple unpredictable pseudo-random number generator", *SIAM Journal on Computing*, vol. 15, 1986, pp. 364–383.
34. BLUM, M., "Three applications of the oblivious transfer: Part I: Coin flipping by telephone; Part II: How to exchange secrets; Part III: How to send certified electronic mail", Department of EECS, University of California, Berkeley, CA, 1981.
35. BLUM, M., "Coin flipping by telephone: A protocol for solving impossible problems", *Proceedings of the 24th IEEE Computer Conference (CompCon)*, 1982, pp. 133–137; reprinted in *SIGACT News*, vol. 15, no. 1, 1983, pp. 23–27.
36. BLUM, M., "How to exchange (secret) keys", *ACM Transactions on Computer Systems*, vol. 1, 1983, pp. 175–193.
37. BLUM, M., "All-or-nothing certified mail", *Workshop on Mathematical Aspects of Cryptography*, Endicott House, MIT, 1985 (no proceedings).
38. BLUM, M., "How to prove a theorem so that no one else can claim it", presented at *International Congress of Mathematicians*, Berkeley, CA, 1986.
39. BLUM, M., FELDMAN, P. and MICALI, S., "Non-interactive zero-knowledge and its applications", *Proceedings of the 20th ACM Symposium on Theory of Computing*, 1988, to appear.
40. BLUM, M. and GOLDWASSER, S., "An efficient probabilistic public-key encryption scheme which hides all partial information", *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 289–299.
41. BLUM, M. and MICALI, S., "How to generate cryptographically strong sequences of pseudo-random bits", *SIAM Journal on Computing*, vol. 13, 1984, pp. 850–864.
42. BLUM, M., VAZIRANI, U. V. and VAZIRANI, V. V., "Reducibility among protocols", *Advances in Cryptology: Proceedings of CRYPTO 83*, Plenum Press, 1984, pp. 137–146.

43. BOHM, D., *Quantum Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1951.
44. BOYAR, J. F., KRETEL, M. W. and KURTZ, S. A., "A discrete logarithm implementation of zero-knowledge blobs", Technical Report 87-002, University of Chicago, 1987.
45. BRASSARD, G., "A note on the complexity of cryptography", *IEEE Transactions on Information Theory*, vol. IT-25, 1979, pp. 232–233.
46. BRASSARD, G., "A time-luck tradeoff in relativized cryptography", *Journal of Computer and System Sciences*, vol. 22, 1981, pp. 280–311.
47. BRASSARD, G., "An optimally secure relativized cryptosystem", *Advances in Cryptography, a Report on CRYPTO 81*, Technical Report no. 82-04, Department of ECE, University of California, Santa Barbara, CA, 1982, pp. 54–58; reprinted in *SIGACT News* vol. 15, no. 1, 1983, pp. 28–33.
48. BRASSARD, G., "On computationally secure authentication tags requiring short secret shared keys", *Advances in Cryptology: Proceedings of CRYPTO 82*, Plenum Press, 1983, pp. 79–86.
49. BRASSARD, G., "Relativized cryptography", *IEEE Transactions on Information Theory*, vol. IT-29, 1983, pp. 877–894.
50. BRASSARD, G., "Cryptology in academia: A ten year retrospective", *Proceedings of the 29th IEEE Computer Conference (CompCon)*, 1987, pp. 222–226.
51. BRASSARD, G., "Cryptology", *Encyclopedia of Mathematics*, vol. 2, D. Reidel Publishing Company, Dordrecht, The Netherlands, 1988.
52. BRASSARD, G. and BRATLEY, P., *Algorithmics: Theory and Practice*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
53. BRASSARD, G., CHAUM, D. and CRÉPEAU, C., "Minimum disclosure proofs of knowledge" (revised version), Technical Report PM-R8710, Centre for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands, 1987.
54. BRASSARD, G., CHAUM, D. and CRÉPEAU, C., "An introduction to minimum disclosure", *CWI Quarterly*, vol. 1, 1988, pp. 3–17.
55. BRASSARD, G. and CRÉPEAU, C., "Zero-knowledge simulation of Boolean circuits", *Advances in Cryptology — CRYPTO '86 Proceedings*, Springer-Verlag, 1987, pp. 224–233.
56. BRASSARD, G. and CRÉPEAU, C., "Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for SAT and beyond", *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, 1986, pp. 188–195.
57. BRASSARD, G., CRÉPEAU, C. and ROBERT, J.-M., "All-or-nothing disclosure of secrets", *Advances in Cryptology — CRYPTO '86 Proceedings*, Springer-Verlag, 1987, pp. 234–238.

58. BRASSARD, G., CRÉPEAU, C. and ROBERT, J.-M., "Information theoretic reductions among disclosure problems", *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, 1986, pp. 168–173.
59. BRICKELL, E. F., "Solving low density knapsacks", *Advances in Cryptology: Proceedings of CRYPTO 83*, Plenum Press, 1984, pp. 25–37.
60. BRICKELL, E. F., "Breaking iterated knapsacks", *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 342–358.
61. BRICKELL, E. F., "The cryptanalysis of knapsack cryptosystems", *Proceedings of the 3rd SIAM Discrete Mathematics Conference*, to appear.
62. BRICKELL, E. F. and ODLYZKO, A. M., "Cryptanalysis: A survey of recent results", *Proceedings of the IEEE*, vol. 76, 1988, to appear.
63. CARTER, J. L. and WEGMAN, M. N., "Universal classes of hash functions", *Journal of Computer and System Sciences*, vol. 18, 1979, pp. 143–154.
64. CEILLIER, R., *La cryptographie*, Que sais-je?, vol. 116 (sic!), Presses Universitaires de France, Paris, France, 1945.
65. CHAUM, D., "Untraceable electronic mail, return addresses and digital pseudonyms", *Communications of the ACM*, vol. 24, 1981, pp. 84–88.
66. CHAUM, D., "Security without identification: Transaction system to make Big Brother obsolete", *Communications of the ACM*, vol. 28, 1985, pp. 1030–1044.
67. CHAUM, D., "Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms", *Advances in Cryptology — EUROCRYPT '85*, Springer-Verlag, 1986, pp. 241–244.
68. CHAUM, D., "Demonstrating that a public predicate can be satisfied without revealing any information about how", *Advances in Cryptology — CRYPTO '86 Proceedings*, Springer-Verlag, 1987, pp. 195–199.
69. CHAUM, D., "The dining cryptographers problem: Unconditional sender and recipient untraceability", *Journal of Cryptology*, vol. 1, 1988, to appear.
70. CHAUM, D., CRÉPEAU, C. and DAMGÅRD, I. B., "Multiparty unconditionally secure protocols", *Proceedings of the 20th ACM Symposium on Theory of Computing*, 1988, to appear.
71. CHAUM, D., DAMGÅRD, I. B. and VAN DE GRAAF, J., "Multiparty computations ensuring privacy of each party's input and correctness of the result", *Advances in Cryptology — CRYPTO '87 Proceedings*, Springer-Verlag, 1988, pp. 87–119.
72. CHAUM, D. and EVERTSE, J.-H., "Cryptanalysis of DES with a reduced number of rounds", *Advances in Cryptology — CRYPTO '85 Proceedings*, Springer-Verlag, 1986, pp. 192–211.

73. CHAUM, D., EVERTSE, J.-H., VAN DE GRAAF, J. and PERALTA, R., "Demonstrating possession of a discrete logarithm without revealing it", *Advances in Cryptology - CRYPTO '86 Proceedings*, Springer-Verlag, 1987, pp. 200–212.
74. CHAUM, D. and SCHAUMÜLLER-BILCH, I. (editors), *Smart Card 2000*, North-Holland, Amsterdam, The Netherlands, 1988.
75. CHOR, B.-Z., GOLDBREICH, O., HASTAD, J., FREIDMANN, J., RUDICH, S. and SMOLENSKY, R., "The bit extraction problem or t -resilient functions", *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, 1985, pp. 396–407.
76. COHEN (BENALOH), J.D. and FISHER, M., "A robust and verifiable cryptographically secure election scheme", *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, 1985, pp. 372–382.
77. COOK, S. A., "The complexity of theorem proving procedures", *Proceedings of the 3rd ACM Symposium on Theory of Computing*, 1971, pp. 151–158.
78. COPPERSMITH, D., "Cheating at mental poker", *Advances in Cryptology - CRYPTO '85 Proceedings*, Springer-Verlag, 1986, pp. 104–107.
79. COPPERSMITH, D., "Cryptography", *IBM Journal of Research and Development*, vol. 31, 1987, pp. 244–248.
80. COPPERSMITH, D., ODLYZKO, A.M. and SCHROEPPLE, R., "Discrete logarithms in $\text{GF}(p)$ ", *Algorithmica*, vol. 1, 1986, pp. 1–15.
81. COUVREUR, C. and QUISQUATER, J.-J., "An introduction to fast generation of large prime numbers", *Philips Journal of Research*, vol. 37, 1982, pp. 231–264; *errata in ibid*, vol. 38, 1983, p. 77.
82. CRÉPEAU, C., "A secure poker protocol that minimizes the effect of player coalitions", *Advances in Cryptology - CRYPTO '85 Proceedings*, Springer-Verlag, 1986, pp. 73–86.
83. CRÉPEAU, C., "A zero-knowledge Poker protocol that achieves confidentiality of the players' strategy, or How to achieve an electronic Poker face", *Advances in Cryptology - CRYPTO '86 Proceedings*, Springer-Verlag, 1987, pp. 239–247.
84. CRÉPEAU, C., "Equivalence between two flavours of oblivious transfers", *Advances in Cryptology - CRYPTO '87 Proceedings*, Springer-Verlag, 1988, pp. 350–354.
85. DAMGÅRD, I. B., *The Application of Claw Free Functions in Cryptography; Unconditional Protection in Cryptographic Protocols*, Ph.D. Thesis, Mathematical Institute, Aarhus University, Aarhus, Denmark, 1988.
86. DAVIDA, G., "Chosen signature cryptanalysis of the RSA (MIT) public key cryptosystem", Technical Report TR-CS-82-2, Department of EECS, University of Wisconsin, Milwaukee, WI, 1982.

87. DAVIES, D.W. and PARKIN, G.I.P., "The average cycle size of the key stream in output feedback encipherment", *Advances in Cryptology: Proceedings of CRYPTO 82*, Plenum Press, 1983, pp. 97–98.
88. DAVIES, D.W. and PRICE, W.L., *Security for Computer Networks: An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer*, John Wiley & Sons, New York, NY, 1984.
89. DAVIO, M., DESMEDT, Y., FOSSEPREZ, M., GOVAERTS, R., HULSBOSCH, J., NEUTJENS, P., PIRET, P., QUISQUATER, J.-J., VANDEWALLE, J. and WOUTERS, P., "Analytical characteristics of the DES", *Advances in Cryptology: Proceedings of CRYPTO 83*, Plenum Press, 1984, pp. 171–202.
90. DAVIO, M., DESMEDT, Y., GOUBERT, J., HOORNAERT, F. and QUISQUATER, J.-J., "Efficient hardware and software implementations of the DES", *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 144–146.
91. DEAVOURS, C.A., "Unicity points in cryptanalysis", *Cryptologia*, vol. 1, 1977, pp. 46–68.
92. DEAVOURS, C.A. and KRUH, L., *Machine Cryptography and Modern Cryptanalysis*, Artech House, Dedham, MA, 1985.
93. DELORME, C. and QUISQUATER, J.-J., "Networks, graphs and security: I. Secrecy on identity of sender and addressee; II. Key exchanges", presented at *First International Conference on Industrial and Applied Mathematics*, Paris, France, June 29 — July 3, 1987; available as Manuscript M210, Philips Research Laboratory, Bruxelles, Belgium, 1987.
94. DENNING, D.E.R., *Cryptography and Data Security*, Addison-Wesley, Reading, MA, 1983.
95. DENNING, D.E.R., "Digital signatures with RSA and other public-key cryptosystems", *Communications of the ACM*, vol. 27, 1984, pp. 388–392.
96. DESMEDT, Y., *Analysis of the Security and New Algorithms for Modern Industrial Cryptography*, Ph.D. Thesis, Katholieke Universiteit Leuven, Belgium, 1984.
97. DESMEDT, Y., "Unconditionally secure authentication schemes and practical and theoretical consequences", *Advances in Cryptology — CRYPTO '85 Proceedings*, Springer-Verlag, 1986, pp. 42–55.
98. DESMEDT, Y. and ODLYZKO, A.M., "A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes", *Advances in Cryptology — CRYPTO '85 Proceedings*, Springer-Verlag, 1986, pp. 516–522.

99. DESMEDT, Y., VANDEWALLE, J. and GOVAERTS, R., "Critical analysis of the security of knapsack public key algorithms", *Abstracts of Papers from IEEE International Symposium on Information Theory*, Les Arcs, France, 1982, pp.115–116; full paper in *IEEE Transactions on Information Theory*, vol. IT-30, 1984, pp.601–611.
100. DIFFIE, W. and HELLMAN, M.E., "Multiuser cryptographic techniques", *Proceedings of AFIPS National Computer Conference*, vol.45, 1976, pp.109–112.
101. DIFFIE, W. and HELLMAN, M.E., "New directions in cryptography", *IEEE Transactions on Information Theory*, vol. IT-22, 1976, pp.644–654.
102. DIFFIE, W. and HELLMAN, M.E., "Exhaustive cryptanalysis of the NBS Data Encryption Standard", *Computer*, vol. 10, June 1977, pp.74–84.
103. DIFFIE, W. and HELLMAN, M.E., "Privacy and authentication: An introduction to cryptography", *Proceedings of the IEEE*, vol. 67, 1979, pp.397–427.
104. EIER, R. and LAGGER, H., "Trapdoors in knapsack cryptosystems", in *Cryptography — Proceedings, Burg Feuerstein 1982*, Lecture Notes in Computer Science, vol. 149, Springer-Verlag, 1983, pp.316–322.
105. ELGAMAL, T., "A public key cryptosystem and a signature scheme based on discrete logarithm", *IEEE Transactions on Information Theory*, vol. IT-31, 1985, pp.469–472.
106. EVANS, A., KANTROWITZ, W. and WEISS, E., "A user authentication scheme not requiring secrecy in the computer", *Communications of the ACM*, vol. 17, 1974, pp.437–442.
107. EVEN, S., GOLDBREICH, O. and LEMPEL, A., "A randomized protocol for signing contracts", *Communications of the ACM*, vol.28, 1985, pp.637–647.
108. EVEN, S. and YACOBI, Y., "Cryptography and NP-completeness", *Proceedings of the 7th International Colloquium on Automata, Languages, and Programming*, Springer-Verlag, 1980, pp.195–207.
109. FEIGE, U., FIAT, A. and SHAMIR, A., "Zero knowledge proofs of identity", *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987, pp.210–217.
110. FEIGENBAUM, J., "Encrypting problem instances, or, . . . , can you take advantage of someone without having to trust him?", *Advances in Cryptology — CRYPTO '85 Proceedings*, Springer-Verlag, 1986, pp.477–488.
111. FEISTEL, H., "Cryptography and Computer Privacy", *Scientific American*, vol. 228, May 1973, pp.15–23.
112. FIAT, A. and SHAMIR, A., "How to prove yourself: Practical solutions to identification and signature problems", *Advances in Cryptology — CRYPTO '86 Proceedings*, Springer-Verlag, 1987, pp.186–194.

113. FORTNOW, L., “The complexity of perfect zero-knowledge”, *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987, pp. 204–209.
114. FORTUNE, S. and MERRIT, M., “Poker protocols”, *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 454–464.
115. FRANKLIN, M., *Mathematical Investigations of the Data Encryption Standard*, Masters Thesis, University of California, Berkeley, CA, 1985.
116. FRIEDMAN, W. F., *Elements of Cryptanalysis*, Aegean Park Press, Laguna Hills, CA, 1976 (written much earlier).
117. GALIL, Z., HABER, S. and YUNG, M., “A private interactive test of a Boolean predicate and minimum-knowledge public-key cryptosystems”, *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, 1985, pp. 360–371.
118. GALIL, Z., HABER, S. and YUNG, M., “Cryptographic computation: Secure fault-tolerant protocols and the public-key model”, *Advances in Cryptology — CRYPTO ’87 Proceedings*, Springer-Verlag, 1988, pp. 135–155.
119. GALLAGER, R. G., *Information Theory and Reliable Communications*, John Wiley & Sons, New York, NY, 1968.
120. GARDNER, M., “A new kind of cipher that would take millions of years to break”, *Scientific American*, vol. 237, August 1977, pp. 120–124.
121. GAREY, M. R. and JOHNSON, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, CA, 1979.
122. GARLINSKI, J., *Intercept: The Enigma War*, J. M. Dent and Sons, Ltd, London, England, 1979.
123. GLASS, A. S., “Could the smart card be dumb?”, *Abstracts of Papers from EUROCRYPT 86*, ISBN 91-7870-077-9, 1986, pp. 1.5A–1.5B.
124. GLEIK, J., “A new approach to protecting secrecy is discovered”, *New York Times*, February 17, 1987, p. C1.
125. GOLDBREICH, O., “A simple protocol for signing contracts”, *Advances in Cryptology: Proceedings of CRYPTO 83*, Plenum Press, 1984, pp. 133–136.
126. GOLDBREICH, O., “Two remarks concerning the Goldwasser-Micali-Rivest signature scheme”, *Advances in Cryptology — CRYPTO ’86 Proceedings*, Springer-Verlag, 1987, pp. 104–110.
127. GOLDBREICH, O., “Towards a theory of software protection and simulation by oblivious RAMs”, *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987, pp. 182–194.
128. GOLDBREICH, O., GOLDWASSER, S. and MICALI, S., “On the cryptographic applications of random functions”, *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 276–288.

129. GOLDBREICH, O., GOLDWASSER, S. and MICALI, S., “How to construct random functions”, *Journal of the ACM*, vol. 33, 1986, pp. 792–807.
130. GOLDBREICH, O., MICALI, S. and WIGDERSON, A., “Proofs that yield nothing but their validity and a methodology of cryptographic protocol design”, *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, 1986, pp. 174–187.
131. GOLDBREICH, O., MICALI, S. and WIGDERSON, A., “How to play any mental game, or: A completeness theorem for protocols with honest majority”, *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987, pp. 218–229.
132. GOLDWASSER, S. and MICALI, S., “Probabilistic encryption and how to play mental poker keeping secret all partial information”, *Proceedings of the 14th ACM Symposium on Theory of Computing*, 1982, pp. 365–377.
133. GOLDWASSER, S. and MICALI, S., “Probabilistic encryption”, *Journal of Computer and System Sciences*, vol. 28, 1984, pp. 270–299.
134. GOLDWASSER, S., MICALI, S. and RACKOFF, C., “The knowledge complexity of interactive proof systems”, *Proceedings of the 17th ACM Symposium on Theory of Computing*, 1985, pp. 291–304; final version to appear in *SIAM Journal on Computing*.
135. GOLDWASSER, S., MICALI, S. and RIVEST, R. L., “A ‘paradoxical’ solution to the signature problem”, *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, 1984, pp. 441–448.
136. GOLDWASSER, S., MICALI, S. and YAO, A. C., “On signatures and authentication”, *Advances in Cryptology: Proceedings of CRYPTO 82*, Plenum Press, 1983, pp. 211–215.
137. GOLDWASSER, S., MICALI, S. and YAO, A. C., “Strong signature schemes”, *Proceedings of the 15th ACM Symposium on Theory of Computing*, 1983, pp. 431–439.
138. GOOD, I. J., “Pioneering work on computers at Bletchley”, in *A History of Computing in the Twentieth Century*, N. Metropolis, J. Howlett and G.-C. Rota (editors), Academic Press, New York, NY, 1980, pp. 31–46; also in *Annals of the History of Computing*, vol. 1, 1979, pp. 38–48.
139. GORDON, J. A., “Strong primes are easy to find”, *Advances in Cryptology: Proceedings of EUROCRYPT 84*, Springer-Verlag, 1985, pp. 216–223.
140. VAN DE GRAAF, J. and PERALTA, R., “A simple and secure way to show the validity of your public key”, *Advances in Cryptology – CRYPTO ’87 Proceedings*, Springer-Verlag, 1988, pp. 128–134.
141. GROLLMANN, J. and SELMAN, A. L., “Complexity measures for public-key cryptosystems”, *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, 1984, pp. 495–503.

142. GUILLOU, L. C., DAVIO, M. and QUISQUATER, J.-J., "Public-key techniques: Randomness and redundancy", *Cryptologia*, vol. XII, 1988, to appear.
143. HASTAD, J., "On using RSA with low exponent in a public key network", *Advances in Cryptology — CRYPTO '85 Proceedings*, Springer-Verlag, 1986, pp. 403–408.
144. HASTAD, J. and SHAMIR, A., "The cryptographic security of truncated linearly related variables", *Proceedings of the 17th ACM Symposium on Theory of Computing*, 1985, pp. 356–362.
145. HELLMAN, M. E., "An extension of the Shannon theory approach to cryptography", *IEEE Transactions on Information Theory*, vol. IT-23, 1977, pp. 289–294.
146. HELLMAN, M. E., "DES will be totally insecure within ten years", *IEEE Spectrum*, vol. 16, July 1979, pp. 32–39.
147. HELLMAN, M. E., "The mathematics of public-key cryptography", *Scientific American*, vol. 241, August 1979, pp. 146–157.
148. HELLMAN, M. E., MERKLE, R. C., SCHROEPPLE, R., WASHINGTON, L., DIFFIE, W., POHLIG, S. and SCHWEITZER, P., "Results on an initial attempt to cryptanalyze the NBS Data Encryption Standard", Technical Report SEL 76-042, Stanford University, 1976.
149. HERZBERG, A. and PINTER, S. S., "Public protection of software", *Advances in Cryptology — CRYPTO '85 Proceedings*, Springer-Verlag, 1986, pp. 158–179.
150. HOORNAERT, F., GOUBERT, J. and DESMEDT, Y., "Efficient hardware implementations of the DES", *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 147–173.
151. DE JONGE, W. and CHAUM, D., "Some variations on RSA signatures & their security", *Advances in Cryptology — CRYPTO '86 Proceedings*, Springer-Verlag, 1987, pp. 49–59.
152. JUENEMAN, R. R., "Analysis of certain aspects of output feedback mode", *Advances in Cryptology: Proceedings of CRYPTO 82*, Plenum Press, 1983, pp. 99–127.
153. KAHN, D., "Modern cryptology", *Scientific American*, vol. 215, July 1966, pp. 38–46.
154. KAHN, D., *The Codebreakers: The Story of Secret Writing*, Macmillan Publishing Co., New York, NY, 1967.
155. KAHN, D., "Cryptology goes public", *Foreign Affairs*, Fall 1979; reprinted in [156], pp. 186–203.
156. KAHN, D., *Kahn on Codes*, Macmillan Publishing Co., New York, NY, 1983.

157. KALISKI, B.S., RIVEST, R.L. and SHERMAN, A.T., “Is the Data Encryption Standard a group? (Results of cycling experiments on DES)”, *Journal of Cryptology*, vol. 1, 1988, to appear.
158. KNUTH, D.E., *The Art of Computer Programming*, vol. 2: *Seminumerical Algorithms*, second edition, Addison-Wesley, Reading, MA, 1981.
159. KOCHANSKI, M., “Developing an RSA chip”, *Advances in Cryptology – CRYPTO ’85 Proceedings*, Springer-Verlag, 1986, pp. 350–357.
160. KOLATA, G.B., “Cryptography: On the brink of a revolution?”, *Science Magazine*, vol. 197, 1977, pp. 747–748.
161. KOLATA, G.B., “New codes coming into use”, *Science Magazine*, vol. 208, 1980, p. 694.
162. KOLATA, G.B., “Cryptographers gather to discuss research”, *Science Magazine*, vol. 214, 1981, pp. 646–647.
163. KONHEIM, A.G., *Cryptography: A Primer*, John Wiley & Sons, New York, NY, 1981.
164. KOTHARI, S., “Generalized linear threshold scheme”, *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 231–241.
165. KRANAKIS E., *Primality and Cryptography*, Wiley-Teubner Series in Computer Science, 1986.
166. LAGARIAS, J.C. and ODLYZKO, A.M., “Solving low-density subset sum problems”, *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, 1983, pp. 1–10.
167. LAKSHMIVARAHAN, S., “Algorithms for public-key cryptosystems: Theory and application”, *Advances in Computers*, vol. 22, 1983, pp. 45–108.
168. LANDAU, S., “Zero knowledge and the department of defense”, *Notices of the American Mathematical Society*, vol. 35, 1988, pp. 5–12.
169. LEMPEL, A., “Cryptology in transition”, *ACM Computing Surveys*, vol. 11, 1979, pp. 285–303.
170. LEVIN, L.A., “One-way functions and pseudo-random generators”, *Proceedings of the 17th ACM Symposium on Theory of Computing*, 1985, pp. 363–365.
171. LIPTON, R., “How to cheat at mental poker”, *Proceedings of AMS short course on Cryptography*, 1981.
172. LONGPRÉ, L., “The use of public-key cryptography for signing checks”, *Advances in Cryptology: Proceedings of CRYPTO 82*, Plenum Press, 1983, pp. 187–197.
173. LUBY, M., MICALI, S. and RACKOFF, C., “How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin”, *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, 1983, pp. 11–21.

174. LUBY, M. and RACKOFF, C., "Pseudo-random permutation generators and cryptographic composition", *Proceedings of the 18th ACM Symposium on Theory of Computing*, 1986, pp. 356—363.
175. MACMILLAN, D., "Single chip encrypts data at 14Mb/s", *Electronics*, June 16, 1981, pp. 161—165.
176. MATYAS, S., "Digital signatures — an overview", *Computer Networks*, vol. 3, 1979, pp. 87—94.
177. MCELIECE, R. J., "A public-key cryptosystem based on algebraic coding theory", DSN Progress Report 42-44, Jet Propulsion Laboratory, 1978, pp. 114—116.
178. McIVOR, R., "Smart cards", *Scientific American*, November 1985, pp. 130—137.
179. MERKLE, R. C., "Secure communications over insecure channels", *Communications of the ACM*, vol. 21, 1978, pp. 294—299.
180. MERKLE, R. C. and HELLMAN, M. E., "Hiding information and signatures in trapdoor knapsacks", *IEEE Transactions on Information Theory*, vol. IT-24, 1978, pp. 525—530.
181. MERKLE, R. C. and HELLMAN, M. E., "On the security of multiple encryption", *Communications of the ACM*, vol. 24, 1981, pp. 465—467.
182. MEYER, C. H. and MATYAS, S. M., *Cryptography: A New Dimension in Computer Data Security*, John Wiley & Sons, New York, NY, 1982.
183. NATIONAL BUREAU OF STANDARDS, "Data Encryption Standard", *Federal Information Processing Standard*, U.S. Department of Commerce, FIPS PUB 46, Washington, DC, 1977.
184. NATIONAL BUREAU OF STANDARDS, "DES modes of operation", *Federal Information Processing Standard*, U.S. Department of Commerce, FIPS PUB 81, Washington, DC, 1980.
185. NEWMAN, D. B., JR. and PICKHOLTZ, R. L., "Cryptography in the private sector", *IEEE Communications Magazine*, vol. 24, 1986, pp. 7—10.
186. ODLYZKO, A. M., "Discrete logarithms in finite fields and their cryptographic significance", *Advances in Cryptology: Proceedings of EUROCRYPT 84*, Springer-Verlag, 1985, pp. 225—314.
187. ORTON, G. A., ROY, M. P., SCOTT, P. A., PEPPARD, L. E. and TAVARES, S. E., "VLSI implementation of public-key encryption algorithms", *Advances in Cryptology — CRYPTO '86 Proceedings*, Springer-Verlag, 1987, pp. 277—301.
188. PELEG, S. and ROSENFELD, A., "Breaking substitution ciphers using a relaxation algorithm", *Communications of the ACM*, vol. 22, 1979, pp. 598—605.
189. PERALTA, R., "Simultaneous security of bits in the discrete log", *Advances in Cryptology — EUROCRYPT '85*, Springer-Verlag, 1986, pp. 62—72.

190. POE, E. A., "The Gold-Bug", *Dollar Newspaper*, June 21 & 28, 1843; reprinted in most anthologies of Poe's tales.
191. POHLIG, S. and HELLMAN, M. E., "An improved algorithm for computing logarithms over $\text{GF}(p)$ and its cryptographic significance", *IEEE Transactions on Information Theory*, vol. IT-24, 1978, pp. 106–110.
192. PURDY, G. P., "A high security log-in procedure", *Communications of the ACM*, vol. 17, 1974, pp. 442–445.
193. QUISQUATER, J.-J. and COUVREUR, C., "Fast decipherment algorithm for RSA public-key cryptosystem", *Electronic Letters*, vol. 18, 1982, pp. 905–907.
194. QUISQUATER, J.-J. and DELVAUX, Y., *Notes sur la cryptographie et la sécurité*, Rapport, Philips Research Laboratory, Bruxelles, Belgium, 1988.
195. RABIN, M. O., "Digital signatures", in *Foundations of Secure Computation*, R. A. DeMillo, D. P. Dobkin, A. K. Jones and R. J. Lipton (editors), Academic Press, New York, NY, 1978, pp. 155–168.
196. RABIN, M. O., "Digital signatures and public-key functions as intractable as factorization", Technical Report MIT/LCS/TR-212, M.I.T., 1978.
197. RABIN, M. O., "Probabilistic algorithm for testing primality", *Journal of Number Theory*, vol. 12, 1980, pp. 128–138.
198. RABIN, M. O., "How to exchange secrets by oblivious transfer", Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.
199. RABIN, M. O., "Transaction protection by beacons", *Journal of Computer and System Sciences*, vol. 27, 1983, pp. 256–267.
200. RANDELL, B., "The COLOSSUS", in *A History of Computing in the Twentieth Century*, N. Metropolis, J. Howlett and G.-C. Rota (editors), Academic Press, New York, NY, 1980, pp. 47–92.
201. REJEWSKI, M., "How Polish mathematicians deciphered the Enigma", *Annals of the History of Computing*, vol. 3, 1981, pp. 213–234.
202. RIVEST, R. L., "A description of a single-chip implementation of the RSA cipher", *LAMBDA Magazine*, vol. 1, 1980, pp. 14–18.
203. RIVEST, R. L., "A short report on the RSA chip", *Advances in Cryptology: Proceedings of CRYPTO 82*, Plenum Press, 1983, p. 327.
204. RIVEST, R. L., "Cryptology", in *Handbook of Theoretical Computer Science*, J. van Leeuwen (managing editor), North-Holland, to appear.
205. RIVEST, R. L., SHAMIR, A. and ADLEMAN, L. M., "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol. 21, 1978, pp. 120–126.

206. RIVEST, R. L. and SHERMAN, A. T., "Randomized encryption techniques", *Advances in Cryptology: Proceedings of CRYPTO 82*, Plenum Press, 1983, pp. 145–163.
207. SEDLAK, H., "The RSA cryptography processor: The first very high speed one-chip solution", *Advances in Cryptology — EUROCRYPT '87*, Springer-Verlag, 1988, to appear.
208. SHAMIR, A., "How to share a secret", *Communications of the ACM*, vol. 24, 1979, pp. 612–613.
209. SHAMIR, A., "A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem", *Proceedings of the 23th IEEE Symposium on Foundations of Computer Science*, 1982, pp. 145–152; full paper in *IEEE Transactions on Information Theory*, vol. IT-30, 1984, pp. 699–704.
210. SHAMIR, A., "On the generation of cryptographically strong pseudo-random sequences", *ACM Transactions on Computer Systems*, vol. 1, 1983, pp. 38–44.
211. SHAMIR, A., "Identity based cryptosystems and signature schemes", *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 47–53.
212. SHAMIR, A., "On the security of the DES", *Advances in Cryptology — CRYPTO '85 Proceedings*, Springer-Verlag, 1986, pp. 280–281.
213. SHAMIR, A., RIVEST, R. L. and ADLEMAN, L., "Mental poker", Technical Report MIT/LCS/TR-125, M.I.T., 1979; reprinted in *Mathematical Gardner*, D. E. Klarner (editor), Wadsworth International, Belmont, CA, 1981, pp. 37–43.
214. SHANNON, C. E., "A mathematical theory of communications", *Bell System Technical Journal*, vol. 27, 1948, pp. 379–423 and 623–656.
215. SHANNON, C. E., "Communication theory of secrecy systems", *Bell System Technical Journal*, vol. 28, 1949, pp. 656–715.
216. SHANNON, C. E., "Prediction and entropy of printed English", *Bell System Technical Journal*, vol. 30, 1951, pp. 50–64.
217. SIMMONS, G. J., "Symmetric and asymmetric encryption", *ACM Computing Surveys*, vol. 11, 1979, pp. 305–330.
218. SIMMONS, G. J., "Verification of treaty compliance — revisited", *Proceedings of the 1983 IEEE Symposium on Security and Privacy*, 1983, pp. 61–66.
219. SIMMONS, G. J., "The prisoners' problem and the subliminal channel", *Advances in Cryptology: Proceedings of CRYPTO 83*, Plenum Press, 1984, pp. 51–67.
220. SIMMONS, G. J., "A secure subliminal channel (?)", *Advances in Cryptology — CRYPTO '85 Proceedings*, Springer-Verlag, 1986, pp. 33–41.
221. SIMMONS, G. J., "Cryptology", *Encyclopaedia Britannica*, Macropaedia, vol. 16.

- 222. SMITH, K., "Watch out hackers, public encryption chips are coming", *Electronic Week*, May 20, 1985, pp. 30—31.
- 223. SOLOVAY, R. and STRASSEN, V., "A fast Monte Carlo test for primality", *SIAM Journal on Computing*, vol. 6, 1977, pp. 84—85; erratum in *ibid*, vol. 7, 1978, p. 118.
- 224. TEDRICK, T., "How to exchange half a bit", *Advances in Cryptology: Proceedings of CRYPTO 83*, Plenum Press, 1984, pp. 147—151.
- 225. TEDRICK, T., "Fair exchange of secrets", *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 434—438.
- 226. VAN TILBORG, H. C. A., *An Introduction to Cryptology*, Kluwer Academic Publishers, Hingham, MA, 1988.
- 227. TOMPA, M., "Zero knowledge interactive proofs of knowledge (a digest)", *Second Conference on Theoretical Aspects of Reasoning about Knowledge*, Monterey, CA, 1988; available as Research Report RC 13282 (#59389), IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY, 1987.
- 228. TOMPA, M. and WOLL, H., "Random self-reducibility and zero-knowledge interactive proofs of possession of information", *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, 1987, pp. 472—482.
- 229. TUCHMAN, W., Talk presented at the *National Computer Conference*, Anaheim, CA, June 1978.
- 230. TUCHMAN, W., "Hellman presents no shortcut solution to the DES", *IEEE Spectrum*, vol. 16, July 1979, pp. 40—41.
- 231. VAZIRANI, U. V. and VAZIRANI, V. V., "Trapdoor pseudo-random number generators with applications to protocol design", *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, 1983, pp. 23—30.
- 232. VAZIRANI, U. V. and VAZIRANI, V. V., "Efficient and secure pseudo-random number generation", *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, 1984, pp. 458—463.
- 233. VERBAUWHEDE, I., HOORNAERT, F., VANDEWALLE, J., DE MAN, H. and GOVAERTS, R., "Security considerations in the design and implementation of a new DES chip", *Advances in Cryptology — EUROCRYPT '87*, Springer-Verlag, 1988, to appear.
- 234. VERNAM, G. S., "Cipher printing telegraph systems for secret wire and radio telegraphic communications", *Journal of the American Institute of Electrical Engineers*, vol. XLV, 1926, pp. 109—115.
- 235. VOLTAIRE, *Dictionnaire philosophique*, 1769.

236. WEGMAN, M. N. and CARTER, J. L., "New hash functions and their use in authentication and set equality", *Journal of Computer and System Sciences*, vol. 22, 1981, pp. 265—279.
237. WIESNER, S., "Conjugate coding", unpublished manuscript circa 1970; subsequently made available in *SIGACT News*, vol. 15, no. 1, 1983, pp. 78—88.
238. WILKES, M. V., *Time-Sharing Computer Systems*, American Elsevier, New York, NY, 1975.
239. WILLIAMS, D. and HINDIN, H. J., "Can software do encryption job?", *Electronics*, July 3, 1980, pp. 102—103.
240. WILLIAMS, H. C., "A modification of the RSA public-key encryption procedure", *IEEE Transactions on Information Theory*, vol. IT-26, 1980, pp. 726—729.
241. WOOTTERS, W. K. and ZUREK, W. H., "A single quantum cannot be cloned", *Nature*, vol. 299, 1982, pp. 802—803.
242. WRIGHT, P., *SpyCatcher*, Viking Penguin Inc., New York, NY, 1987.
243. YAO, A. C.-C., "Theory and applications of trapdoor functions", *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, 1982, pp. 80—91.
244. YAO, A. C.-C., "Protocols for secure computations", *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, 1982, pp. 160—164.
245. YAO, A. C.-C., "How to generate and exchange secrets", *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, 1986, pp. 162—167.
246. YARDLEY, H. O., *The American Black Chamber*, Bobbs Merrill, Indianapolis, IN, 1931; reprinted by Ballantine Books, New York, NY.
247. YIU, K. and PETERSON, K., "A single-chip VLSI implementation of the discrete exponential public key distribution system", *Proceedings of IEEE Global Telecommunications Conference*, vol. 1, 1982, pp. 173—179.
248. YUNG, M., "Cryptoprotocols: Subscription to a public key, the secret blocking and the multi-player mental poker game", *Advances in Cryptology: Proceedings of CRYPTO 84*, Springer-Verlag, 1985, pp. 439—453.
249. YUNG, M., "A secure and useful keyless cryptosystem", *Information Processing Letters*, vol. 21, 1985, pp. 35—38.
250. ZORPETTE, G., "Breaking the enemy's code", *IEEE Spectrum*, September 1987, pp. 47—51.