Preface

The art of war teaches us to rely not on the likelihood of the enemy's not coming, but on our own readiness to receive him; not on the chance of his not attacking, but rather on the fact that we have made our position unassailable.

Sun Tzu
The Art of War (500 BC)

The book is about the cryptanalytic attacks on RSA. RSA is the first *workable* and *practical* public-key cryptographic system, invented in 1977 and published in 1978, by Rivest, Shamir and Adleman, then all at the Massachusetts Institute of Technology (MIT), and is still the most widely used cryptographic systems in e.g., online transactions, emails, smartcards, and more generally electronic and mobile commerce over the Internet, for which its three inventors received the year 2002 Turing Award, a prize considered to be the equivalent *Nobel Prize for Computer Science*. The security of RSA relies on the computational intractability of the Integer Factorization Problem (IFP), for which, no efficient (i.e., polynomial-time) algorithm is known. To get an idea how difficult the *integer factorization* is, let us consider the following 2048 bits (617 digits) composite number, known as RSA-2048:

251959084756578934940271832400483985714292821262040320277771378360_
436620207075955562640185258807844069182906412495150821892985591491_
761845028084891200728449926873928072877767359714183472702618963750_
149718246911650776133798590957000973304597488084284017974291006424_
586918171951187461215151726546322822168699875491824224336372590851_
418654620435767984233871847744479207399342365848238242811981638150_
106748104516603773060562016196762561338441436038339044149526344321_
901146575444541784240209246165157233507787077498171257724679629263_
863563732899121548314381678998850404453640235273819513786365643912_
120103971228221207203 57.

It is a *product* of two *prime numbers*. The RSA Data Security Incorporation currently offers a $200,000 prize for the first person or group finding its two prime factors. The basic idea of RSA encryption and decryption is, surprisingly, rather simple:

$$C \equiv M^e \pmod{N}, \quad M \equiv C^d \pmod{N},$$

where $N = pq$ with p and q prime, M, C, e and d are the plaintext, ciphertext, encryption exponent and decryption exponent, respectively. Note that e and d must be satisfied with the condition that $ed \equiv 1 \pmod{\phi(N)}$, where $\phi(N) = (p-1)(q-1)$ is Euler's ϕ-function. Let, for example, $e = 65537$, N be the above mentioned number RSA-2048, and C the following number:

218598056144555493024019389629177159753811144728543422921500499254_
181211032562087679022259831067991286101190897695119357754765408522_
697956638242922870637083231694404873947694078432775781998614979942_
064361669462614088852741600217233052059574880668463536030287944235_
822627708134997061064700771693064600712629809165416998449992925313_
374281387325903328781863209595468701560742767599157207314869432305_
892651836189508103764678721683360183118994273706398707795480800698_
501878875875150532123738006235671958527639461339868604410378449818_
383913059864587128396200112815989134558427750667427151537609736712_
04647757116059031684587.

To recover M from C one requires to find d; to find d one needs to calculate $\phi(N)$; to calculate $\phi(N)$ one needs to factor N. But unfortunately, factorizing N is intractable when N is large (in the present case, N is a 2048-bit number, which is far beyond the computing power of any factoring algorithm on any computer at present); no polynomial-time factoring algorithm is known so far. Thus, RSA is secure and C is safe since it is difficult to recover M from C without factoring N. This is essentially the whole idea of RSA! One can try to decrypt the above given RSA ciphertext C or try to factor the number RSA-2048 in order to get an idea how difficult it is to break RSA or to factor a large number.

The book consists of ten chapters. Chapter 1 presents some computational and mathematical preliminaries, particularly the theory and practice of tractable and intractable computations in number theory. Chapter 2 introduces the basic concepts and theory of the RSA cryptographic system and its variants in a broad sense. As the security of RSA is based on the intractability of the Integer Factorization Problem (IFP), which is also closely related to the Discrete Logarithm Problem (DLP), the attacks based on solutions to IFP problem are discussed in Chapter 3, whereas the attacks based on solutions to DLP problem are discussed in Chapter 4. As quantum algorithm is applicable to both the IFP problem and the DLP problem, Chapter 5 will discuss some quantum attacks on RSA via quantum order finding, quantum factoring and quantum discrete logarithm solving. Chapter 6 concentrates on some simple elementary number-theoretic attacks on RSA, including e.g., forward attack, short plaintext attack, common modulus attack and fixed-point attack. It is common that to speed-up the computation of RSA encryption, a short public exponent e is often used. It is also true for the RSA decryption if a short private exponent d is used. However, the use of short exponent e or d can be dangerous. So, in Chapter 7 we shall discuss some cryptanalytic attacks on the short RSA public exponent e, whereas in Chapter 8 we shall discuss some attacks on the short RSA private exponent d. In Chapter 9, a completely different type of attacks, namely, the side-channel attacks on RSA, are discussed. Unlike the mathematical/algorithmic attacks in the previous chapters, side-channel attacks do not exploit the mathematical properties or weakness of the RSA algorithm/system itself, but exploit the hardware implementation issues of the system. In other words, these attacks are nothing to do with the RSA algorithm/system itself but have something to do with the hardware implementation of the RSA algorithm/system. Chapter 10, the final chapter, presents some quantum resistant, non-factoring based cryptographic systems as an alternative/replacement to RSA, such as lattice based and code-based cryptosystems, so that once RSA is proved to be insecure, there is an immediate replacement to the insecure RSA.

The book is self-contained and the materials presented in the book have been extensively classroom tested for various courses in Cryptography and Cryptanalysis at Aston and Coventry Universities in England, and the South China University of Technology and Nankai University in China. Many parts of the materials in the book have also been presented in seminars in various universities around the world. Hence, the book is suitable *either* as a research reference for public-key cryptology in general and for RSA cryptology in particular, *or* as a graduate text in the field.

Acknowledgments

The author would like to thank Prof Sushil Jajodia of George Mason University, USA, Prof Glyn James and Dr Anne James of Coventry University, UK, Prof Stephen Cook of the University of Toronto, Canada, and Prof Richard Brent of Oxford University and Australian National University for their encouragement, support and help during the writing of the book. Special thanks must also be given to Susan Lagerstrom-Fife and Sharon Palleschi, the editors at Springer in Boston, USA, for their encouragement, support and help.

Parts of book were written while the author visited the following three places: the Department of Computer Science at the University of Toronto (UT) in March-April 2005, hosted by Prof Stephen Cook and supported by UT and the Royal Society London, the Mathematical Science Institute at the Australian National University (ANU) in October-November 2006, hosted by Prof Richard Brent and supported by ANU and the Royal Society London, and the Department of Mathematics at the Massachusetts Institute of Technology (MIT) in July-Sept 2007, hosted by Prof Michael Sipser and supported by MIT.

Special thanks must also be given to Prof Glyn James at Coventry University and Prof Richard Brent at Oxford University and Australian National University for reading the whole manuscript of the book, to Prof Brain Scotney at Ulster University for his constant encouragement during the writing of the book, and to Prof Michael Sipser for inviting me to visit and work at MIT where the book was finally completed.

The struggle between code-makers and code-breakers is endless. The struggle between attacks and anti-attacks on RSA is also endless as soon as RSA is till in use. New ideas and new attacks on RSA may be conceived and invented anytime. So comments, corrections and suggestions on the book, and new ideas and news attacks on RSA are particularly very welcome from the readers, and can be sent to any one of my following three email addresses: song.yan@beds.ac.uk, syan@math.mit.edu, or syan@cs.toronto.edu, so that I can incorporate them into a future edition of the book. Thank you for your help in advance.

Cambridge, Massachusetts, August 2007 S. Y. Y.

2. RSA Public-Key Cryptography

The increased use of shared communications channels, particularly wireless and local area networks (LAN's), leads to greater connectivity, but also to a much greater opportunity to intercept data and forge messages, $\cdots$ The only practical way to maintain privacy and integrity of information is by using public-key cryptography.

Peter Wegner
Professor of Computer Science, Brown University

2.1 Introduction

Cryptography (from the Greek *Kryptós*, "hidden" or "secret", and gráphein, "writing") is the study of the processes of encryption (mapping the original message, called the plaintext, into a secret form, called the the ciphertext, using the encryption key), and decryption (inverting the ciphertext back to the plaintext, using the corresponding decryption key), in such a way that only the intended recipients can decrypt and read the original messages.

$$\text{Cryptograpgy} \overset{\text{def}}{=} \text{Encryption} \oplus \text{Decryption}$$

The methods of encryption are often also called ciphers. Cryptanalysis (from the Greek *Kryptós* and *analýein*, "loosing"), on the other hand, is the study of breaking the encryptions without the knowledge of the key:

$$\text{Cryptanalysis} \overset{\text{def}}{=} \text{Cryptanalytic Attacks on Encryptions}$$

Cryptology (from the Greek *Kryptós* and lógos, "word") consists of both cryptography and cryptanalysis:

$$\text{Cryptology} \overset{\text{def}}{=} \text{Cryptography} \oplus \text{Cryptanalysis}$$

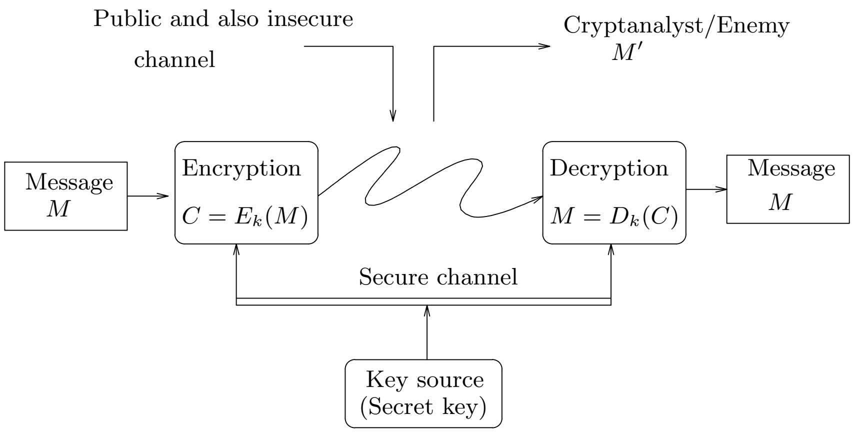


Figure 2.1. Cryptography and Cryptanalysis

The idea of encryption, decryption, cryptanalysis and secure communications over an insecure channel, usually a computer network particularly the Internet, can be depicted as in Figure 2.1. Throughout the book, we shall assume that Bob sends a message to Alice, but Eve wants to cryptanalyze the message:

$$\text{Bob} \underset{\substack{\downarrow \\ \text{Eve}}}{\xrightarrow{\text{Message}}} \text{Bob}. \tag{2.1}$$

Modern cryptography, however, is the study of the mathematical systems of encryption and decryption, to solve the security, particularly the network security problems as follows:

(1) *Confidentiality* or *privacy*: To stop Eve to understand Bob's message to Alice even if she can intercept and get the message.

(2) *Integrity*: To make sure that Bob's message has not been modified by Eve.

(3) *Authentication* or *authorization*: To make sure the message received by Alice is indeed from Bob, not from Eve.

(4) *Non-repudiation*: To stop Bob later to deny the sending of his message. Non-repudiation is particularly important in electronic commerce since we need to make sure that a consumer cannot deny the authorization of a purchase. It must be noted that however, in some applications such as in electronic voting, the non-repudiation feature should, in fact, be avoided, since the voter does not want to disclose the authorization of a vote regardless whether of not he actually did the vote.

Such a mathematical system is called the *cryptographic system*, or *cryptosystems* for short.

Definition 2.1.1. A conventional *secret-key cryptosystem* (or *secret-key encryption*, or *secret-key cipher*) $\mathcal{S}$ may be formally defined as follows (depicted in Figure 2.2):

$$\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, M, C, k, E, D) \tag{2.2}$$

where

(1) $\mathcal{M}$ is the set of plaintexts, called the plaintext space.

(2) $\mathcal{C}$ is the set of cipherexts, called the ciphertext space.

(3) $\mathcal{K}$ is the set of keys, called the key space.

(4) $M \in \mathcal{M}$ is a piece of plaintext.

(5) $C \in \mathcal{C}$ is a piece of ciphertext.

(6) $k \in \mathcal{K}$ is the key for both encryption and decryption.

(7) E is the encryption function

$$E_k : \ M \mapsto C$$

where $M \in \mathcal{M}$ maps to $C \in \mathcal{C}$, using the key k, such that

$$C = E_k(M) \tag{2.3}$$

(8) D is the decryption function

$$D_k : \ C \mapsto M$$

where $C \in \mathcal{C}$ maps to $M \in \mathcal{M}$, using the *same* key k again such that

$$M = D_k(C) \tag{2.4}$$

satisfying

$$E_k D_k = 1 \text{ and } D_k(C) = D_k(E_k(M)) = M. \tag{2.5}$$

Cryptanalysis, on the other hand, is the study of the *cryptanalytic attacks* on cryptosystems, aiming at breaking the cryptosystems without using/knowing the keys, but according to the *Kerckhoff principle*, the cryptanalyst who wants to break the cryptosystem knows the cryptosystem. For example, the following is a ciphertext presented by Édouard Lucas at the 1891 meeting of the French Association for Advancement of Science (see page 388 of Williams [332]), based on Étienne Bazeries' cylindrical cryptography (see pages 244–250 of Kahn [161]); it has never been decrypted, and hence is suitable as a good challenge to the interested reader:

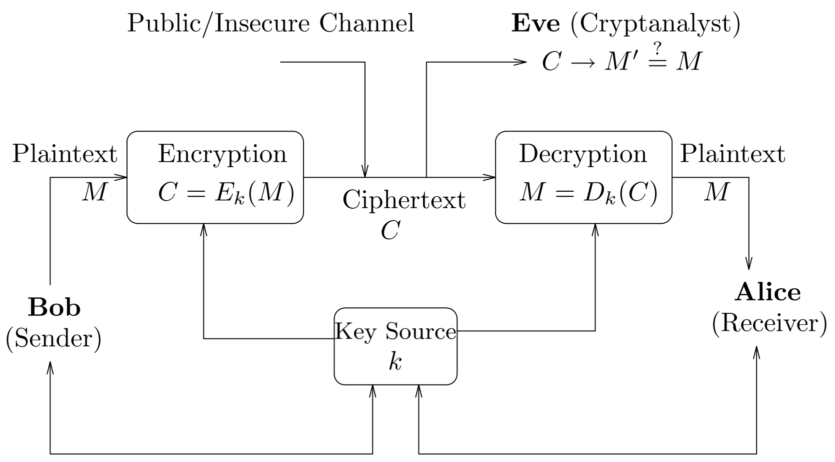


Figure 2.2. Secret-Key Cryptography

XSJOD	PEFOC	XCXFM	RDZME
JZCOA	YUMTZ	LTDNJ	HBUSQ
XTFLK	XCBDY	GYJKK	QBSAH
QHXPE	DBMLI	ZOYVQ	PRETL
TPMUK	XGHIV	ARLAH	SPGGP
VBQYH	TVJYJ	NXFFX	BVLCZ
LEFXF	VDMUB	QBIJV	ZGGAI
TRYQB	AIDEZ	EZEDX	KS

The *security* or the *unbreakability* of any cryptographic system is of paramount importance. There are several different types of security measures for a cryptographic system:

(1) *Unconditionally secure*: A cryptosystem is unconditionally secure if a cryptanalyst cannot determine how to find the plaintext M regardless of how much ciphertext C and computer time/resources he has available to him. A one-time pad (OTP) can be shown to be unconditionally secure, as the key is used only for one time (i.e., there are at least as many keys as the plaintexts), the key string is a random string, and the key size is at least as long as the plaintext string. Unconditional security for cryptosystems is called *perfect secrecy*, or *information-theoretic security*. A cryptosystem $\mathcal{S}$ is *unconditionally unbreakable* if $\mathcal{S}$ is unconditionally secure. In general, cryptosystems do not offer perfect secrecy, in particular, public-key cryptosystems, such as the RSA cryptosystem described in next sections, cannot be unconditionally secure/breakable since once

a ciphertext C is given, its corresponding plaintext M can in principle be recovered by computing all possible plaintexts until C is obtained, an attack called forward search, which will be discussed later. Nevertheless, unconditionally unbreakable cryptosystem exists; it was first proved by Shannon in his 1949 seminar paper in modern cryptography "Communication Theory of Secrecy Systems" [285]. Thus the prominent English mathematician J. E. Littlewood (1885–1977) commented:

The legend that every cipher is breakable is of course absurd, though still widespread among people who should know better.

(2) *Computationally secure*: A cryptosystem $\mathcal{S}$ is computationally secure or *polynomially secure* if a cryptanalyst cannot decrypt C to get M in polynomial-time (or space). A cryptosystem $\mathcal{S}$ is *computationally unbreakable*, if it is unbreakable in polynomial-time, that is, it is computationally secure. According to the Cook-Karp thesis, any problem that can not be solved in polynomial-time is *computationally infeasible*, thus, if the cryptanalytic attack on a cryptosystem is computationally infeasible, then the cryptosystem is computationally secure and computationally unbreakable. There are several types of computationally security:

(2-1) *Provably secure*: A cryptosystem $\mathcal{S}$ is *provably secure* if the difficulty of breaking it can be shown to be essentially as difficult as solving a well-known and supposedly difficult mathematical problems such as the integer factorization problem IFP or the discrete logarithm problem DLP. For example, the Rabin cryptosystem described later is provably secure, as the security of the Rabin cryptosystem is equivalent to the IFP problem.

(2-2) *Practical/conjectured secure*: A cryptosystem $\mathcal{S}$ is *practical secure* if the breaking of the system $\mathcal{S}$ is *conjectured* as difficult as solving a well-known and supposedly difficult mathematical problems such as the integer factorization problem IFP or the discrete logarithm problem DLP. For example, breaking the most popular public-key cryptosystem RSA is conjectured as hard as solving the IFP problem, but so far this has never been proved or disproved. Most of the public-key and secret-key cryptosystems in current use are in this type.

There are several types of possible cryptanalytic attacks on a cryptosystem $\mathcal{S}$, depending on what information the cryptanalyst might already have regarding $\mathcal{S}$:

(1) *Ciphertext-only attack*: Only a piece of ciphertext C is known to the cryptanalyst whose goal is to find the corresponding plaintext M and/or the key k. This is the most difficult type of attack; any cryptosystem vulnerable to this type of attack is considered to be *completely* insecure.

(2) *Known-plaintext attack*: The cryptanalyst has a piece of plaintext M and the corresponding ciphertext C. The goal is the find the key k so that other ciphertexts using the same encryption/key may be decrypted.

(3) *Chosen-plaintext attack*: The cryptanalyst has gained temporary access to the encryption machinery, so he can choose a piece of plaintext M and construct the corresponding ciphertext C. The goal here is to find the key k.

(4) *Chosen-ciphertext attack*: The cryptanalyst has gained temporary access to the decryption machinery, so can choose a piece of ciphertext C and construct the corresponding plaintext M. The goal here is also to find the key k.

A good cryptosystem should resist all of these types of attacks, so that it is impossible for a cryptanalysis to get the key k or to find the plaintext M in polynomial-time.

Remark 2.1.1. Public-key cryptosystems, such as the RSA cryptosystem described in the next sections, give rise to the chosen-ciphertext attack, since the cryptanalyst may specify/obtain some ciphertext using the public-key and learn the corresponding plaintext. In fact, all public-key cryptographic systems are vulnerable to a chosen-ciphertext attack, which, however, can be avoided by adding appropriate redundancy or randomness (padding or salting) prior to encryption.

2.2 Public-Key Cryptography

Surprisingly, *public-key cryptography*, or *asymmetric key cryptography* (see Figure 2.3), is almost the same (although the idea is different) as the *secret-key cryptography*, or *symmetric key cryptography*, except that the keys k for encryption and decryption are different. That is, we need two keys, e_k and d_k, such that e_k is used for encryption and d_k for decryption, respectively. As e_k is only used for encryption, it can be made public; only d_k must be kept a secret for decryption. To distinguish public-key cryptosystems from secret-key cryptosystems, e_k is called the *public key*, and d_k the *private key*; only the key used in secret-key cryptosystems is called the *secret key*. Remarkably enough, secret-key cryptography has a very long history, almost as long as our human civilization; whereas public-key cryptography has a rather short history. In fact the official date of birth of public-key cryptography is 1976, when Diffie and Hellman, then both at Stanford University, published their seminal paper *New Directions in Cryptography* [101] (see the first page of the paper in Figure 2.4). It is in this seminal paper that they *first* publicly proposed the completely new idea of public-key cryptography as well as digital signatures. Although Diffie and Hellman did not have a practical implementation of their idea, they did propose [101] an alternative key-exchange scheme over the

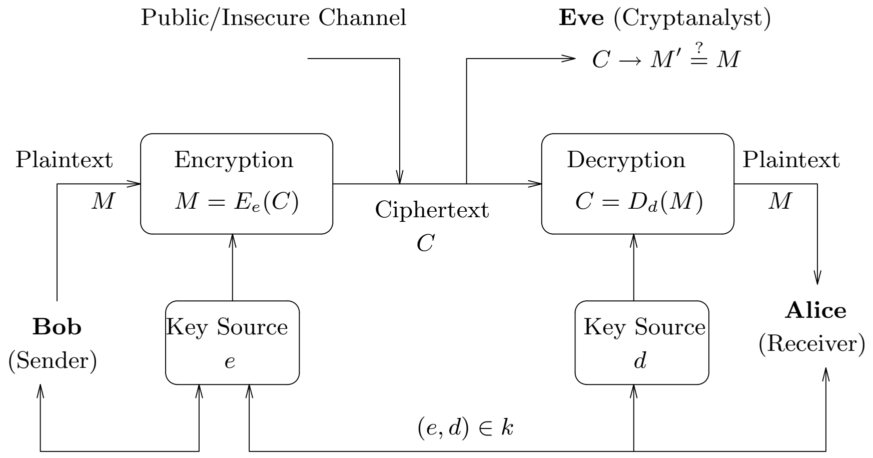
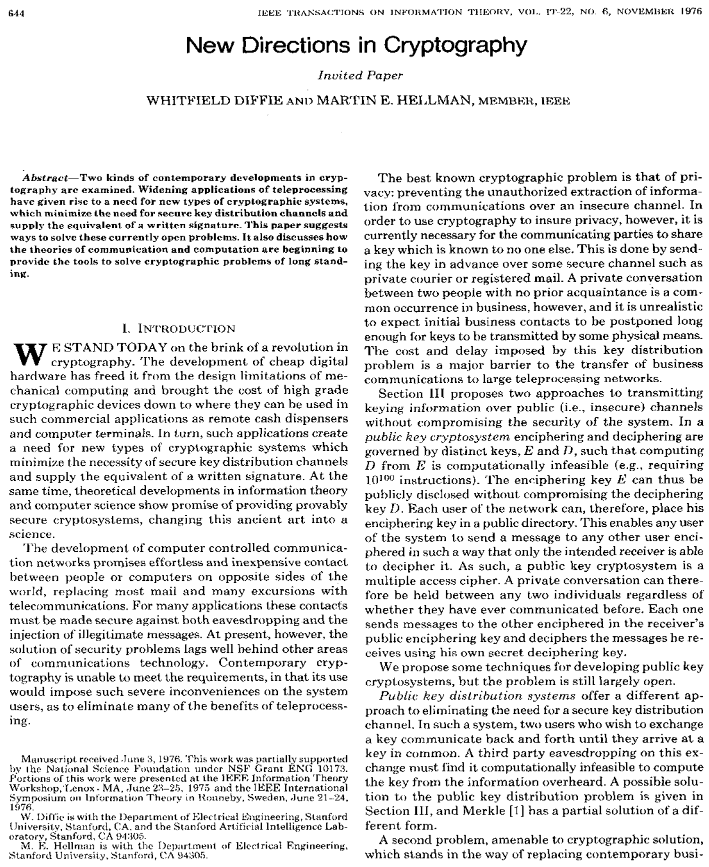


Figure 2.3. Public-Key Cryptography

insecure channel, based on the intractability of the DLP problem and using some of the ideas proposed earlier (although published later) by Merkle [208] (published in 1978, but submitted in 1975; see the first page of this paper in Figure 2.5).

Shortly after the publication of Diffie and Hellman's paper, Rivest, Shamir and Adleman, then all at Massachusetts Institute of Technology (MIT), proposed a first workable and practical public-key cryptosystem in in 1977 [262] (see the first page of the paper in Figure 2.6). The system is now known as RSA; it was first made public to the world and became famous probably because Gardner's 1978 paper in Scientific American [115].

It is interesting to note that the British cryptographers Ellis, Cocks and Williamson at the UK Government's Communications-Electronics Security Group/Government Communications Headquarters (CESG/GCHQ) also claimed that they secretly discovered the public-key encryption years before the US scientists. There are of course two different universes of cryptography: public (particularly for people working in academic institutions) and secret (particularly for people working for militaries and governments). Ellis-Cocks-Williamson certainly deserved some credit for their contribution to the development of public-key cryptography. It should be noted that Hellman and his colleagues not only invented the public-key encryption, but also the digital signatures which had not been mentioned in any of Ellis-Cocks-Williamson's documents/papers.



644 IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

***Abstract*—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.**

I. INTRODUCTION

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

The development of computer controlled communication networks promises effortless and inexpensive contact between people or computers on opposite sides of the world, replacing most mail and many excursions with telecommunications. For many applications these contacts must be made secure against both eavesdropping and the injection of illegitimate messages. At present, however, the solution of security problems lags well behind other areas of communications technology. Contemporary cryptography is unable to meet the requirements, in that its use would impose such severe inconveniences on the system users, as to eliminate many of the benefits of teleprocessing.

Manuscript received June 3, 1976. This work was partially supported by the National Science Foundation under NSF Grant ENG 10173. Portions of this work were presented at the IEEE Information Theory Workshop, Lenox, MA, June 23–25, 1975 and the IEEE International Symposium on Information Theory in Ronneby, Sweden, June 21–24, 1976.

W. Diffie is with the Department of Electrical Engineering, Stanford University, Stanford, CA, and the Stanford Artificial Intelligence Laboratory, Stanford, CA 94305.

M. E. Hellman is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305.

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channels without compromising the security of the system. In a *public key cryptosystem* enciphering and deciphering are governed by distinct keys, E and D, such that computing D from E is computationally infeasible (e.g., requiring 10^{100} instructions). The enciphering key E can thus be publicly disclosed without compromising the deciphering key D. Each user of the network can, therefore, place his enciphering key in a public directory. This enables any user of the system to send a message to any other user enciphered in such a way that only the intended receiver is able to decipher it. As such, a public key cryptosystem is a multiple access cipher. A private conversation can therefore be held between any two individuals regardless of whether they have ever communicated before. Each one sends messages to the other enciphered in the receiver's public enciphering key and deciphers the messages he receives using his own secret deciphering key.

We propose some techniques for developing public key cryptosystems, but the problem is still largely open.

Public key distribution systems offer a different approach to eliminating the need for a secure key distribution channel. In such a system, two users who wish to exchange a key communicate back and forth until they arrive at a key in common. A third party eavesdropping on this exchange must find it computationally infeasible to compute the key from the information overheard. A possible solution to the public key distribution problem is given in Section III, and Merkle [1] has a partial solution of a different form.

A second problem, amenable to cryptographic solution, which stands in the way of replacing contemporary busi-

Figure 2.4. The First Page of Diffie and Hellman's Paper

The implementation of public-key cryptosystems is based on *trapdoor one-way functions.*

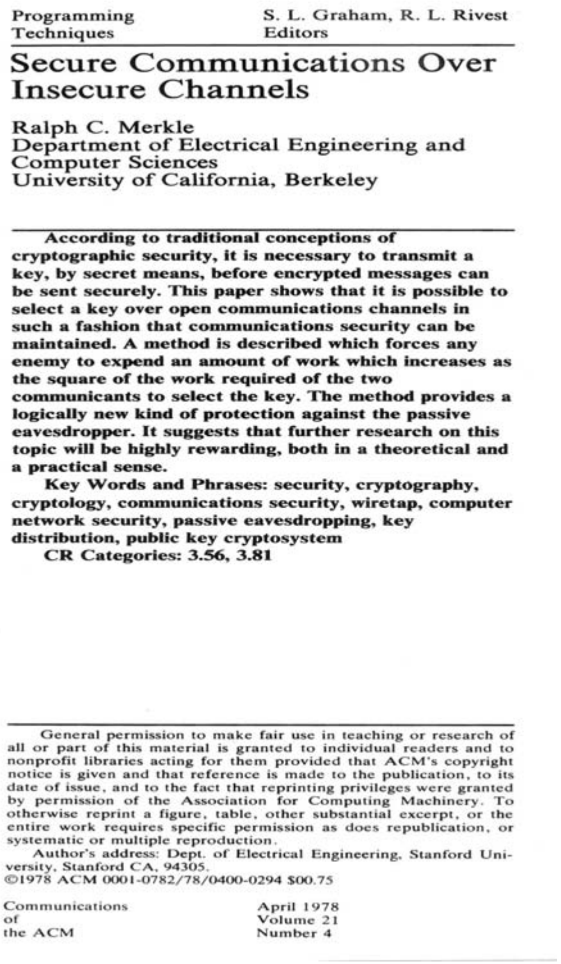
Definition 2.2.1. Let S and T be finite sets. A *one-way function*

$$f: S \to T \tag{2.6}$$

is an invertible function satisfying

(1) f is easy to compute, that is, given $x \in S$, $y = f(x)$ is easy to compute.

(2) f^{-1}, the inverse function of f, is difficult to compute, that is, given $y \in T$, $x = f^{-1}(y)$ is difficult to compute.



Programming Techniques — S. L. Graham, R. L. Rivest, Editors

Secure Communications Over Insecure Channels

Ralph C. Merkle
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

According to traditional conceptions of cryptographic security, it is necessary to transmit a key, by secret means, before encrypted messages can be sent securely. This paper shows that it is possible to select a key over open communications channels in such a fashion that communications security can be maintained. A method is described which forces any enemy to expend an amount of work which increases as the square of the work required of the two communicants to select the key. The method provides a logically new kind of protection against the passive eavesdropper. It suggests that further research on this topic will be highly rewarding, both in a theoretical and a practical sense.

Key Words and Phrases: security, cryptography, cryptology, communications security, wiretap, computer network security, passive eavesdropping, key distribution, public key cryptosystem

CR Categories: 3.56, 3.81

General permission to make fair use in teaching or research of all or part of this material is granted to individual readers and to nonprofit libraries acting for them provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery. To otherwise reprint a figure, table, other substantial excerpt, or the entire work requires specific permission as does republication, or systematic or multiple reproduction.

Author's address: Dept. of Electrical Engineering, Stanford University, Stanford CA, 94305.

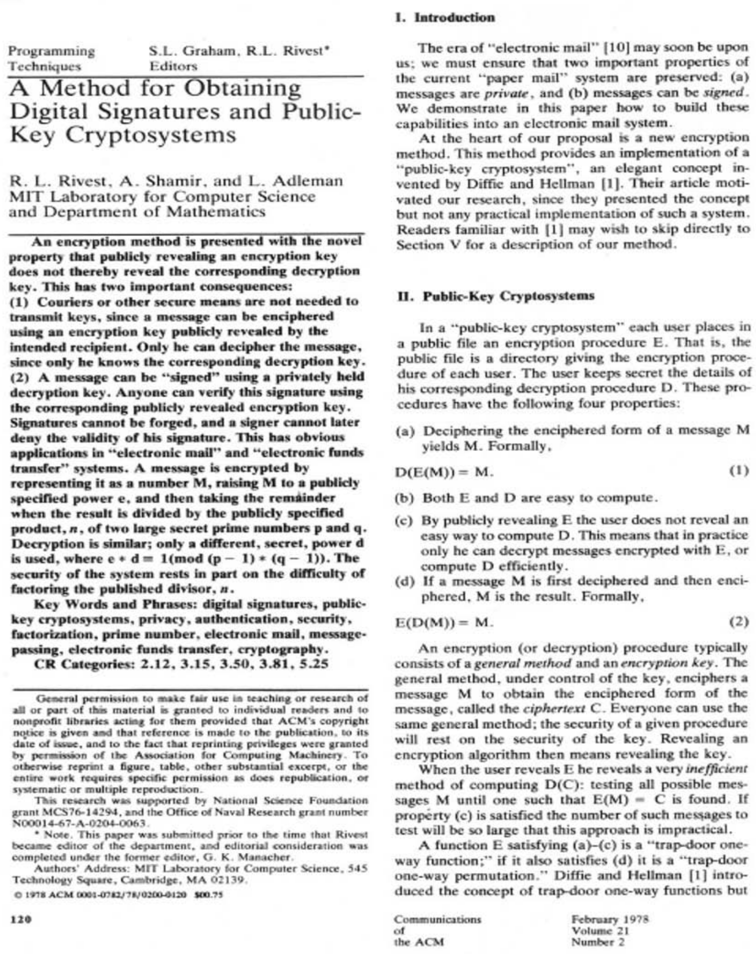
©1978 ACM 0001-0782/78/0400-0294 $00.75

Communications of the ACM — April 1978, Volume 21, Number 4

Figure 2.5. The First Page of Merkle's Paper

(3) f^{-1} is easy to compute when a trapdoor (i.e., a secret string of information associated with the function) becomes available.

A function f satisfying only the first two conditions is also called a one-to-one one-way function. If f satisfies further the third condition, it is called a *trapdoor one-way function*.



Programming Techniques — S.L. Graham, R.L. Rivest* Editors

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R. L. Rivest, A. Shamir, and L. Adleman
MIT Laboratory for Computer Science
and Department of Mathematics

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:
(1) Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intended recipient. Only he can decipher the message, since only he knows the corresponding decryption key.
(2) A message can be "signed" using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious applications in "electronic mail" and "electronic funds transfer" systems. A message is encrypted by representing it as a number M, raising M to a publicly specified power e, and then taking the remainder when the result is divided by the publicly specified product, *n*, of two large secret prime numbers p and q. Decryption is similar; only a different, secret, power d is used, where e * d ≡ 1(mod (p − 1) * (q − 1)). The security of the system rests in part on the difficulty of factoring the published divisor, *n*.

Key Words and Phrases: digital signatures, public-key cryptosystems, privacy, authentication, security, factorization, prime number, electronic mail, message-passing, electronic funds transfer, cryptography.

CR Categories: 2.12, 3.15, 3.50, 3.81, 5.25

General permission to make fair use in teaching or research of all or part of this material is granted to individual readers and to nonprofit libraries acting for them provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery. To otherwise reprint a figure, table, other substantial excerpt, or the entire work requires specific permission as does republication, or systematic or multiple reproduction.

This research was supported by National Science Foundation grant MCS76-14294, and the Office of Naval Research grant number N00014-67-A-0204-0063.

* Note. This paper was submitted prior to the time that Rivest became editor of the department, and editorial consideration was completed under the former editor, G. K. Manacher.

Authors' Address: MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139.

© 1978 ACM 0001-0782/78/0200-0120 $00.75

I. Introduction

The era of "electronic mail" [10] may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem", an elegant concept invented by Diffie and Hellman [1]. Their article motivated our research, since they presented the concept but not any practical implementation of such a system. Readers familiar with [1] may wish to skip directly to Section V for a description of our method.

II. Public-Key Cryptosystems

In a "public-key cryptosystem" each user places in a public file an encryption procedure E. That is, the public file is a directory giving the encryption procedure of each user. The user keeps secret the details of his corresponding decryption procedure D. These procedures have the following four properties:

(a) Deciphering the enciphered form of a message M yields M. Formally,

D(E(M)) = M. (1)

(b) Both E and D are easy to compute.

(c) By publicly revealing E the user does not reveal an easy way to compute D. This means that in practice only he can decrypt messages encrypted with E, or compute D efficiently.

(d) If a message M is first deciphered and then enciphered, M is the result. Formally,

E(D(M)) = M. (2)

An encryption (or decryption) procedure typically consists of a *general method* and an *encryption key*. The general method, under control of the key, enciphers a message M to obtain the enciphered form of the message, called the *ciphertext* C. Everyone can use the same general method; the security of a given procedure will rest on the security of the key. Revealing an encryption algorithm then means revealing the key.

When the user reveals E he reveals a very *inefficient* method of computing D(C): testing all possible messages M until one such that E(M) = C is found. If property (c) is satisfied the number of such messages to test will be so large that this approach is impractical.

A function E satisfying (a)–(c) is a "trap-door one-way function;" if it also satisfies (d) it is a "trap-door one-way permutation." Diffie and Hellman [1] introduced the concept of trap-door one-way functions but

120 — Communications of the ACM — February 1978 Volume 21 Number 2

Figure 2.6. The First Page of RSA's Paper

Example 2.2.1. The following functions are one-way functions:

(1) $f:\ pq \mapsto n$ is a one-way function, where p and q are prime numbers. The function f is easy to compute since the multiplication of p and q can be done in polynomial time. However, the computation of f^{-1}, the inverse of f is hard (this is the IFP problem).

(2) $f:\ x \mapsto g^x \bmod N$ is a one-way function. The function f is easy to compute since the modular exponentiation $g^x \bmod N$ can be performed in polynomial time. But the computation of f^{-1}, the inverse of f is hard (this is the DLP problem).

(3) $f:\ x \mapsto x^k \bmod N$ is a trapdoor one-way function, where $N = pq$ with p and q primes, and $kk' \equiv 1 \pmod{\phi(N)}$. It is obvious that f is easy

to compute since the modular exponentiation $x^k \bmod N$ can be done in polynomial time, but f^{-1}, the inverse of f (i.e., the kth root of x modulo N) is difficult to compute. However, if k', the trapdoor is given, f can be easily inverted, since $(x^k)^{k'} = x$.

Now we are in a position to introduce the formal definition of public-key cryptography.

Definition 2.2.2. A public-key cryptosystem $\mathcal{CS}$ may be formally defined as follows:

$$\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, M, C, e, d, E, D) \tag{2.7}$$

where

(1) $\mathcal{M}$ is the set of plaintexts, called the plaintext space.
(2) $\mathcal{C}$ is the set of cipherexts, called the ciphertext space.
(3) $\mathcal{K}$ is the set of keys, called the key space.
(4) $M \in \mathcal{M}$ is a piece of particular plaintext.
(5) $C \in \mathcal{C}$ is a piece of particular ciphertext.
(6) $e \neq d$ and $(e, d) \in \mathcal{K}$ is the key.
(7) E is the encryption function

$$E_{e_k} : \; M \mapsto C$$

where $M \in \mathcal{M}$ maps to $C \in \mathcal{C}$, using the public-key e_k, such that

$$C = E_{e_k}(M) \tag{2.8}$$

(8) D is the decryption function

$$D_{d_k} : \; C \mapsto M$$

where $C \in \mathcal{C}$ maps to $M \in \mathcal{M}$, using the private-key d_k such that

$$M = D_{d_k}(C) \tag{2.9}$$

satisfying

$$E_{e_k} D_{d_k} = 1 \text{ and } D_{d_k}(C) = D_{d_k}(E_{e_k}(M)) = M. \tag{2.10}$$

The main task in public-key cryptography is to find a suitable trap-door one-way function, so that both encryption and decryption are easy to perform for authorized users, whereas decryption, the inverse of the encryption, should be computationally infeasible for an unauthorized user.

2.3 RSA Public-Key Cryptography

This section introduces the basic idea and theory of the most popular and widely-used public-key cryptosystem RSA.

Definition 2.3.1. The *RSA public-key cryptosystem* may be formally defined as follows (Depicted in Figure 2.7):

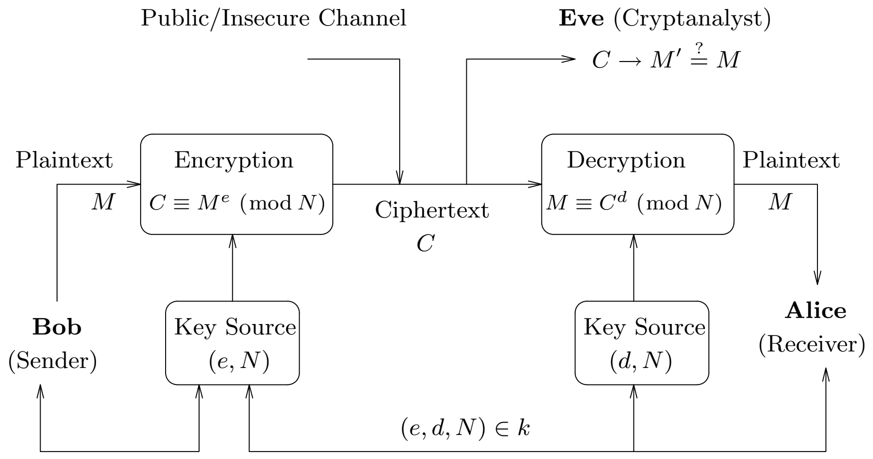


Figure 2.7. RSA Public-Key Cryptography

$$\text{RSA} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, M, C, e, d, N, E, D) \tag{2.11}$$

where

(1) $\mathcal{M}$ is the set of plaintexts, called the plaintext space.
(2) $\mathcal{C}$ is the set of cipherexts, called the ciphertext space.
(3) $\mathcal{K}$ is the set of keys, called the key space.
(4) $M \in \mathcal{M}$ is a piece of particular plaintext.
(5) $C \in \mathcal{C}$ is a piece of particular ciphertext.
(6) $N = pq$ is the modulus with p, q prime numbers, usually each with at least 100 digits.
(7) $\{(e, N), (d, N)\} \in \mathcal{K}$ with $e \neq d$ are the encryption and encryption keys, respectively, satisfying

$$ed \equiv 1 \pmod{\phi(N)} \tag{2.12}$$

where $\phi(N) = (p-1)(q-1)$ is the Euler ϕ-function and defined by $\phi(N) = \#(\mathbb{Z}_N^*)$, the number of elements in the multiplicative group $\mathbb{Z}_N^*$.

(8) E is the encryption function

$$E_{e,N}: \ M \mapsto C$$

That is, $M \in \mathcal{M}$ maps to $C \in \mathcal{C}$, using the public-key (e, N), such that

$$C \equiv M^e \pmod{N}. \tag{2.13}$$

(9) D is the decryption function

$$D_{d,N}: \ C \mapsto M$$

That is, $C \in \mathcal{C}$ maps to $M \in \mathcal{M}$, using the private-key (d, N), such that

$$M \equiv C^d \equiv (M^e)^d \pmod{N}. \tag{2.14}$$

The idea of RSA can be best depicted in Figure 2.8.

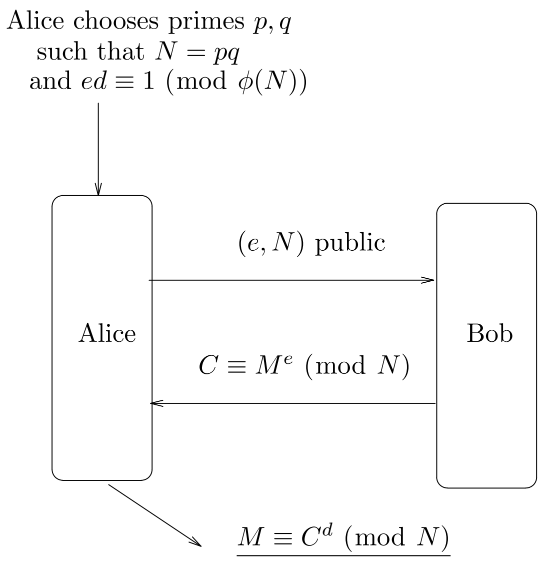


Figure 2.8. RSA Encryption and Decryption

Theorem 2.3.1 (The Correctness of RSA). Let M, C, N, e, d be plaintext, ciphertext, encryption exponent, decryption exponent, and modulus, respectively. Then

$$(M^e)^d \equiv M \pmod{N}.$$

Proof. Notice first that

$$\begin{aligned}
C^d &\equiv (M^e)^d \pmod N && \text{(since } C \equiv M^e \pmod N\text{)}\\
&\equiv M^{1+k\phi(N)} \pmod N && \text{(since } ed \equiv 1 \pmod{\phi(N)}\text{)}\\
&\equiv M \cdot M^{k\phi(N)} \pmod N\\
&\equiv M \cdot (M^{\phi(N)})^k \pmod N\\
&\equiv M \cdot (1)^k \pmod N && \text{(by Euler's Theorem } a^{\phi(n)} \equiv 1 \pmod N\text{)}\\
&\equiv M
\end{aligned}$$

The result thus follows. □

Both encryption $C \equiv M^e \pmod N$ and decryption $M \equiv C^d \pmod N$ of RSA can be implemented in polynomial-time by Algorithm 1.3.5. For example the RSA encryption can be implemented as follows:

Algorithm 2.3.1. Given (e, M, N), this algorithm finds $C \equiv M^e \pmod N$, or given (d, C, N), finds $M \equiv C^d \pmod N$ in time polynomial in $\log e$ or $\log d$, respectively.

```
Given $(e, M, N)$ to find $C$
Set $C \leftarrow 1$
While $e \geq 1$ do
        if $e \bmod 2 = 1$
          then $C \leftarrow C \cdot M \bmod N$
        $M \leftarrow M^2 \bmod N$
        $e \leftarrow \lfloor e/2 \rfloor$
Print C
```

```
Given $(d, C, N)$ to find $M$
Set $M \leftarrow 1$
While $d \geq 1$ do
        if $d \bmod 2 = 1$
          then $M \leftarrow M \cdot C \bmod N$
        $C \leftarrow C^2 \bmod N$
        $d \leftarrow \lfloor d/2 \rfloor$
Print M
```

Remark 2.3.1. For the decryption process in RSA, as the authorized user knows d and hence knows p and q, thus instead of directly working on $M \equiv C^d \pmod N$, he can speed-up the computation by working on the following two congruences:

$$\begin{aligned}
M_p &\equiv C^d \equiv C^{d \bmod p-1} \pmod p\\
M_q &\equiv C^d \equiv C^{d \bmod q-1} \pmod q
\end{aligned}$$

and then use the Chinese Remainder Theorem to get

$$M \equiv M_p \cdot q \cdot q^{-1} \bmod p + M_q \cdot p \cdot p^{-1} \bmod q \pmod N. \tag{2.15}$$

The Chinese Remainder Theorem is a two-edged sword. On the one hand, it provides a good way to speed-up the computation/performance of the RSA decryption, which can even be easily implemented by a low-cost crypto-chip [129]. On the other hand, it may introduce some serious security problems vulnerable to some side-channel attacks, particularly the random fault attacks; we shall discuss this in Section 8.4.

Example 2.3.1. Let the letter-digit encoding be as follows:

$$\text{space} = 00, A = 01, B = 02, \cdots, Z = 26.$$

(We will use this digital representation of letters throughout the book.) Let also

$$\begin{array}{rcl} e &=& 9007, \\ M &=& 200805001301070903002315180419000118050019172105011309\_ \\ && 190800151919090618010705, \\ N &=& 114381625757888867669235779976146612010218296721242362\_ \\ && 562561842935706935245733897830597123563958705058989075\_ \\ && 147599290026879543541. \end{array}$$

Then the encryption can be done by using Algorithm 2.3.1:

$$\begin{array}{rcl} C &\equiv& M^e \\ &\equiv& 968696137546220614771409222543558829057599911245743198\_ \\ && 746951209308162982251457083569314766228839896280133919\_ \\ && 90551829945157815154 \pmod{N}. \end{array}$$

For the decryption, since the two prime factors p and q of N are known to the authorized person who does the decryption:

$$\begin{array}{rcl} p &=& 349052951084765094914784961990389813341776463849338784\_ \\ && 3990820577, \\ q &=& 327691329932667095499619881908344614131776429679929425\_ \\ && 39798288533, \end{array}$$

then

$$\begin{array}{rcl} d &\equiv& 1/e \\ &\equiv& 106698614368578024442868771328920154780709906633937862\_ \\ &\equiv& 801226224496631063125911774470873340168597462306553968\_ \\ &\equiv& 544513277109053606095 \pmod{(p-1)(q-1)}. \end{array}$$

Thus, the original plaintext M can be recovered either directly by using Algorithm 2.3.1, or indirectly by a combined use of Algorithm 2.3.1 and the Chinese Remainder Theorem (2.15):

$$\begin{array}{rcl} M &\equiv& C^d \\ &=& 200805001301070903002315180419000118050019172105011309\_ \\ && 190800151919090618010705 \pmod{N} \end{array}$$

which is “THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE”.

Remark 2.3.2. Prior to RSA, Pohlig and Hellman in 1978 [241] proposed a secret-key cryptography based on arithmetic modulo p, rather than $N = pq$. The Pohlig-Hellman system works as follows: Let M and C be the plain and cipher texts, respectively. Choose a prime p, usually with more than 200 digits, and a secret encryption key e such that $e \in \mathbb{Z}^+$ and $e \leq p-2$. Compute $d \equiv 1/e \pmod{(p-1)}$. (e, p) and of course d must be kept as a secret.

[1] **Encryption:**

$$C \equiv M^e \pmod{p}. \tag{2.16}$$

This process is easy for the authorized user:

$$\{M, e, p\} \xrightarrow[\text{easy}]{\text{find}} \{C \equiv M^e \pmod{p}\}. \tag{2.17}$$

[2] **Decryption:**

$$M \equiv C^d \pmod{p}. \tag{2.18}$$

For the authorized user who knows (e, p), this process is easy, since d can be easily computed from e.

[3] **Cryptanalysis:** The security of this system is based on the infeasibility of the Discrete Logarithm Problem. For example, for a cryptanalyst who does not know e or d would have to compute:

$$e \equiv \log_M C \pmod{p}.$$

Remark 2.3.3. One of the most important features of RSA encryption is that it can also be used for digital signatures. Let M be a document to be signed, and $N = pq$ with p, q primes, (e, d) the public and private exponents as in RSA encryption scheme. Then the processes of RSA signature signing and signature verification are just the same as that of the decryption and encryption; that is use d for signature signing and e signature verification as follows (see also Figure 2.9):

[1] **Signature signing**:

$$S \equiv M^d \pmod{N} \tag{2.19}$$

The signing process can only be done by the authorized person who has the private exponent d.

[2] **Signature verification**:

$$M \equiv S^e \pmod{N} \tag{2.20}$$

This verification process can be done by anyone since (e, N) is public.

Of course, RSA encryption and RSA signature can be used together to obtain a signed encrypted document to be sent over an insecure network.

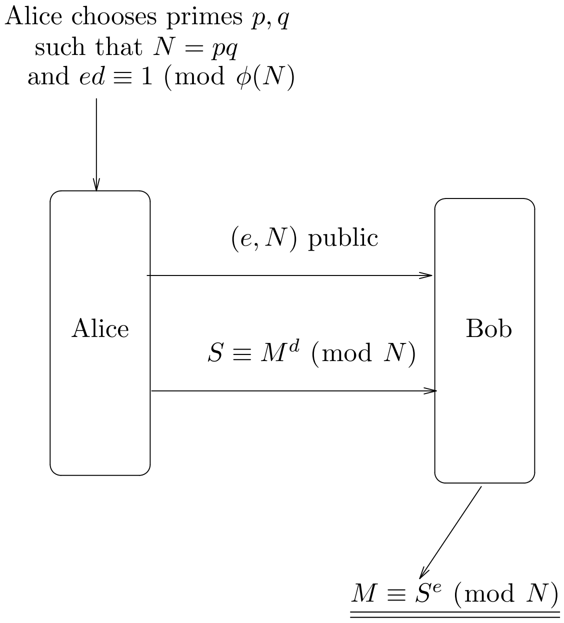


Figure 2.9. RSA Digital Signature

2.4 RSA Problem and RSA Assumption

As can be seen from the previous section, the whole idea of the RSA encryption and decryption is as follows:

$$\left.\begin{array}{rcl} C & \equiv & M^e \pmod{N}, \\ M & \equiv & C^d \pmod{N} \end{array}\right\} \tag{2.21}$$

where

$$\left.\begin{array}{rcl} ed & \equiv & 1 \pmod{\phi(N)} \\ N & = & pq \text{ with } p, q \in \text{ Primes.} \end{array}\right\} \tag{2.22}$$

Thus, the *RSA function* can be defined by

$$f_{\text{RSA}} : M \mapsto M^e \bmod N. \tag{2.23}$$

The *inverse of the RSA function* is then defined by

$$f_{\text{RSA}}^{-1} : M^e \mapsto M \bmod N. \tag{2.24}$$

Clearly, the RSA function is a *one-way trap-door function*, with

$$\{d, p, q, \phi(N)\} \tag{2.25}$$

the RSA *trap-door information*mitrap-door information. For security purposes, this set of information must be kept as a secret and should never be disclosed in anyway even in part. Now suppose that Bob sends C to Alive, but Eve intercepts it and wants to understand it. Since Eve only has (e, N, C) and does not have any piece of the trap-door information in (2.25), then it should be infeasible/intractable for her to recover M from C:

$$\{e, N, C \equiv M^e \pmod N\} \xrightarrow{\text{hard}} \{M \equiv C^d \pmod N\}. \tag{2.26}$$

On the other hand, for Alice, since she knows d, which implies that she knows all the pieces of trap-door information in (2.25), since

$$\{d\} \overset{\mathcal{P}}{\Longleftrightarrow} \{p\} \overset{\mathcal{P}}{\Longleftrightarrow} \{q\} \overset{\mathcal{P}}{\Longleftrightarrow} \{\phi(N)\} \tag{2.27}$$

We shall explain the relations in (2.27) in Chapter 6). Thus, it is easy for Alice to recover M from C:

$$\{N, C \equiv M^e \pmod N\} \xrightarrow[\text{easy}]{\{d,p,q,\phi(N)\}} \{M \equiv C^d \pmod N\}. \tag{2.28}$$

Why is it hard for Eve to recover M from C? This is because Eve is facing a hard computational problem, namely, the *RSA problem* [264]:

The RSA problem: Given the RSA public-key (e, N) and the RSA ciphertext C, find the corresponding RSA plaintext M. That is,

$$\{e, N, C\} \longrightarrow \{M\}.$$

It is conjectured although it has never been proved or disproved that:

The RSA conjecture: Given the RSA public-key (e, N) and the RSA ciphertext C, it is hard to find the corresponding RSA plaintext M. That is,

$$\{e, N, C\} \xrightarrow{\text{hard}} \{M\}.$$

But how hard is it for Alice to recover M from C? This is another version of the RSA conjecture, often called the *RSA assumption*, which again has never been proved or disproved:

The RSA assumption: Given the RSA public-key (e, N) and the RSA ciphertext C, then finding M is as hard as factoring the RSA modulus N. That is,

$$\text{IFP}(N) \Longleftrightarrow \text{RSA}(M)$$

provided that N is sufficiently large and randomly generated, and M and C are random integers between 0 and $N-1$. More precisely, it is conjectured (or assumed) that

$$\text{IFP}(N) \overset{\mathcal{P}}{\Longleftrightarrow} \text{RSA}(M).$$

That is, if N can be factorized in polynomial-time, then M can be recovered from C in polynomial-time. In other words, cryptoanalyzing RSA must be as difficult as solving the IFP problem. But the problem is, as we discussed previously, that no one knows whether or not IFP can be solved in polynomial-time, so RSA is only assumed to be secure, not proved to be secure:

$$\text{IFP}(N) \text{ is hard } \longrightarrow \text{RSA}(M) \text{ is secure.}$$

The real situtaion is that

$$\text{IFP}(N) \overset{\surd}{\Longrightarrow} \text{RSA}(M),$$

$$\text{IFP}(N) \overset{?}{\Longleftarrow} \text{RSA}(M).$$

Now we can return to answer the question that how hard is it for Alice to recover M from C? By the RSA assumpition, cryptanalyzing C is as hard as factoring N. The fastest known integer factorization algorithm, the Number Field Sieve (NFS), runs in time

$$\mathcal{O}(\exp(c(\log N)^{1/3}(\log\log N)^{2/3}))$$

where $c = (64/9)^{1/3}$ if a general version of NFS, GNFS, is used for factoring an arbitrary integer N whereas $c = (32/9)^{1/3}$ if a special version of NFS, SNFS, is used for factoring a special form of integer N. As in RSA, the modululs $N = pq$ is often chosen be a large general composite integer $N = pq$ with p and q the same bit size, which makes SNFS is not useful. This means that RSA cannot be broken in polynomial-time, but in subexponential-time, which makes RSA secure, again, by assumption. Thus, readers should note that the RSA problem is *assumed* to be *hard*, and the RSA cryptosystem is *conjectured* to be *secure* .

2.5 RSA-Type Crytposystems

RSA is a cryptographic system based on factoring as its security relies on the intractability of the Integer Factorization Problem (IFP). However, RSA is not the only cryptographic system based on factoring. There are in fact many cryptographic systems whose security depends on the intractability of the IFP problem. In this section, we study some of these systems. If the RSA problem can be solved in polynomial-time, all the factoring based cryptographic systems can be broken in polynomial-time. Thus, we could regard all the factoring based cryptographic systems as various variants of the RSA cryptographic systems, or *RSA-type cryptosystems.* In a more broad sense, as IFP is related to DLP/ECDLP, all DLP/ECDLP-based cryptosystems may also be regarded as RSA-type cryptosystems.

Rabin's M^2 Cryptoystem

As can be seen from the previous sections, RSA uses M^e for encryption, with $e \geq 3$ (3 is the smallest possible public exponent), we might call RSA encryption M^e encryption. In 1979, Michael Rabin proposed a scheme based on M^2 encryption. rather than the M^e for $e \geq 3$ encryption used in RSA. A brief description of the Rabin system is as follows (see also Figure 2.10).

[1] **Key generation:** Let $N = pq$ with p, q odd primes satisfying

$$p \equiv q \equiv 3 \pmod{4}. \tag{2.29}$$

[2] **Encryption:**

$$C \equiv M^2 \pmod{N}. \tag{2.30}$$

[3] **Decryption:** Use the Chinese Remainder Theorem to solve the system of congruences:

$$\left\{\begin{array}{l} M_p \equiv \sqrt{C} \pmod{p} \\ M_q \equiv \sqrt{C} \pmod{q} \end{array}\right. \tag{2.31}$$

to get the four solutions: $\{\pm M_p, \pm M_q\}$. The true plaintext M will be one of these four values.

[4] **Cryptanalysis:** A cryptanalyst who can factor N can compute the four square roots of C modulo N, and hence can recover M from C. Thus, breaking the Rabin system is equivalent to factoring N.

Unlike the RSA cryptosystem whose security was only conjectured to be equivalent to the intractability of IFP, the security of Rabin-Williams is proved to be equivalent to the intractability of IFP. First notice that there is a fast algorithm to compute the square roots modulo N if $N = pq$ is known.

Consider the following quadratic congruence

$$x^2 \equiv y \pmod{p} \tag{2.32}$$

there are essentially three cases for the prime p:

(1) $p \equiv 3 \pmod{4}$,

(2) $p \equiv 5 \pmod{8}$,

(3) $p \equiv 1 \pmod{8}$.

All three cases may be solved by the following process:

$$\left\{\begin{array}{ll} \text{if } p \equiv 3 \pmod{4}, & x \equiv \pm y^{\frac{p+1}{4}} \pmod{p}, \\ \text{if } p \equiv 5 \pmod{8}, & \left\{\begin{array}{ll} \text{if } y^{\frac{p+1}{4}} = 1, & x \equiv \pm y^{\frac{p+3}{8}} \pmod{p} \\ \text{if } y^{\frac{p+1}{4}} \neq 1, & x \equiv \pm 2y(4y)^{\frac{p-5}{8}} \pmod{p}. \end{array}\right. \end{array}\right. \tag{2.33}$$

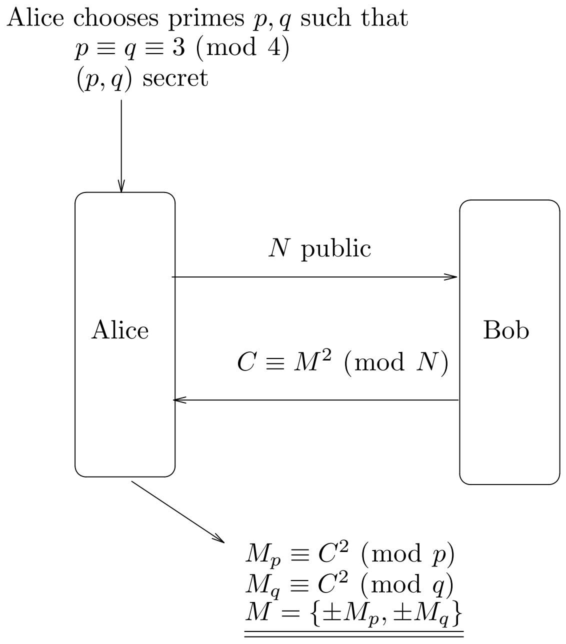


Figure 2.10. Rabin System

Algorithm 2.5.1 (Computing Square Roots Modulo pq). Let $N = pq$ with p and q odd prime and $y \in \mathrm{QR}_N$. This algorithm will find all the four solutions in x to congruence $x^2 \equiv y \pmod{pq}$ in time $\mathcal{O}((\log p)^4)$.

[1] Use (2.33) to find a solution r to $x^2 \equiv y \pmod p$.

[2] Use (2.33) to find a solution s to $x^2 \equiv y \pmod q$.

[3] Use the Extended Euclid's algorithm to find integers c and d such that $cp + dq = 1$.

[4] Compute $x \equiv \pm(rdq \pm scp) \pmod{pq}$.

On the other hand, if there exists an algorithm to find the four solutions in x to $x^2 \equiv y \pmod N$, then there exists an algorithm to find the prime factorization of N. The following is the algorithm.

Algorithm 2.5.2 (Factoring via Square Roots). This algorithm seeks to find a factor of N by using an existing square root finding algorithm (namely, Algorithm 2.5.1).

[1] Choose at random an integer x such that $\gcd(x, N) = 1$, and compute $x^2 \equiv a \pmod N$.

[2] Use Algorithm 2.5.1 to find four solutions in x to $x^2 \equiv a \pmod N$.

[3] Choose one of the four solutions, say y such that $y \not\equiv \pm x \pmod{N}$, then compute $\gcd(x \pm y, N)$.

[4] If $\gcd(x \pm y, N)$ reveals p or q, then go to step [5], or otherwise, go to step [1].

[5] Exit.

Theorem 2.5.1. Let $N = pq$ with p, q odd prime. If there exists a polynomial-time algorithm A to factor $N = pq$, then there exists an algorithm B to find a solution to $x^2 \equiv y \pmod{N}$, for any $y \in \mathrm{QR}_N$.

Proof. If there exists an algorithm A to factor $N = pq$, then there exists an algorithm (in fact, Algorithm 2.5.1), which determines $x = \pm(rdq \pm scp) \pmod{pq}$, as defined in Algorithm 2.5.1, for $x^2 \equiv y \pmod{N}$. Clearly, Algorithm 2.5.1 runs in polynomial-time. □

Theorem 2.5.2. Let $N = pq$ with p, q odd prime. If there exists a polynomial-time algorithm A to find a solution to $x^2 \equiv a \pmod{N}$, for any $a \in \mathrm{QR}_N$, then there exists a probabilistic polynomial time algorithm B to find a factor of N.

Proof. First note that for N composite, x and y integer, if $x^2 \equiv y^2 \pmod{N}$ but $x \not\equiv \pm y \pmod{N}$, then $\gcd(x + y, N)$ are proper factors of n. If there exists an algorithm A to find a solution to $x^2 \equiv a \pmod{N}$ for any $a \in \mathrm{QR}_N$, then there exists an algorithm (in fact, Algorithm 2.5.2), which uses algorithm A to find four solutions in x to $x^2 \equiv a \pmod{N}$ for a random x with $\gcd(x, N) = 1$. Select one of the solutions, say, $y \not\equiv \pm x \pmod{N}$, then by computing $\gcd(x \pm y, N)$, the probability of finding a factor of N will be $\geq 1/2$. If Algorithm 2.5.2 runs for k times and each time randomly chooses a different x, then the probability of not factoring N is $\leq 1/2^k$. □

So, finally, we have

Theorem 2.5.3. Factoring integers, computing the modular square roots, and breaking the Rabin cryptosystem are computationally (deterministic polynomial-time) equivalent. That is,

$$\mathrm{IFP}(N) \overset{\mathcal{P}}{\Longleftrightarrow} \mathrm{Rabin}(M). \tag{2.34}$$

Williams' Improved M^2 Cryptoystem

Williams [327] proposed a modified version of the RSA cryptographic system, particularly the Rabin's M^2 system in order to make it suitable as a public-key encryption scheme (Rabin's original system was intended to be used as a digital signature scheme). A description of *Williams' M^2 encryption* is as follows (suppose Bob wishes to send Alice a ciphertext $C \equiv M^2 \pmod{N}$):

[1] **Key generation:** Let $N = pq$ with q and q primes such that

$$\left.\begin{array}{l} p \equiv 3 \pmod 8 \\ q \equiv 7 \pmod 8 \end{array}\right\}$$

So, $N \equiv 5 \pmod 8$ and $(N, 2)$ is used public-key. The private-key d is defined by

$$d = \frac{(p-1)(q-1)}{4} + 1$$

[2] **Encryption:** Let $\mathcal{M}$ be plaintext space containing all possible plaintexts M such that

$$2(2M+1) < N \text{ if the Jacobi symbol } \left(\frac{2M+1}{N}\right) = -1$$

$$4(2M+1) < N \text{ if the Jacobi symbol } \left(\frac{2M+1}{N}\right) = 1$$

The first step in encryption is for all $M \in \mathcal{M}$, put

$$M' = E_1(M) = \begin{cases} 2(2M+1) \text{ if the Jacobi symbol } \left(\frac{2M+1}{N}\right) = -1 \\ 4(2M+1) \text{ if the Jacobi symbol } \left(\frac{2M+1}{N}\right) = 1. \end{cases}$$

The last step in encryption is just the same as Rabin's encryption:

$$C \equiv (M')^2 \pmod N$$

[3] **Decryption:** On the reverse order of the encryption, the first step in decryption is as follows:

$$C' = D_2(C) \equiv C^d \pmod N$$

and the last step in decryption is defined by:

$$M = D_1(C') = \begin{cases} \frac{\frac{M'}{4}-1}{2} & \text{if } M' \equiv 0 \pmod 4 \\ \frac{\frac{N-M'}{4}-1}{2} & \text{if } M' \equiv 1 \pmod 4 \\ \frac{\frac{M'}{2}-1}{2} & \text{if } M' \equiv 2 \pmod 4 \\ \frac{\frac{N-M'}{2}-1}{2} & \text{if } M' \equiv 3 \pmod 4). \end{cases}$$

The whole process of encryption and decryption is as follows:

$$M \xrightarrow{E_1} M' \xrightarrow{E_2} C \xrightarrow{D_2} M' \xrightarrow{D_1} M.$$

[4] **Cryptanalysis:** A cryptanalyst who can factor N can find d, and hence can recover M from C. Thus, breaking the Williams' system is equivalent to factoring N.

Theorem 2.5.4 (Correctness of Williams' M^2 encryption). Let $M \in \mathcal{M}$. Then

$$M = D_1(D_2(E_2(E_1(M)))).$$

Theorem 2.5.5 (Equivalence of Williams(M) and IFP(N)). Breaking Williams' M^2 encryption (i.e., finding M from C) is equivalent to factoring the modulus N. That is,

$$\text{IFP(N)} \overset{\mathcal{P}}{\Longleftrightarrow} \text{Williams}(M). \tag{2.35}$$

For the justification od the above two theorem, see Williams [327].

Just the same as Rabin's system, Williams' M^2 encryption is also provably secure, as breaking the Williams' $M^2 \bmod N$ encryption is equivalent to factoring N, where the N is a special form of $N = pq$, with p, q primes and $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$. Note that this special integer factorization problem is not the same as the general IFP problem, although there is no any known reason to believe this special factoring problem is any easier than the general factoring problem. But unlike Rabin's system, Williams' M^2 encryption can be easily generalized to the general M^e encryption with $e > 2$, as in RSA. Thus, Williams' M^2 encryption is not just a variant of Rabin system, but also a variant of the general RSA system. In fact, Williams' original paper [327] discussed the general case that

$$ed \equiv \frac{\frac{(p-1)(q-1)}{4} + 1}{2} \ (\text{mod} \lambda(N)).$$

Williams' M^2 encryption improved Rabin's M^2 encryption by eliminating the 4 : 1 ciphertext ambiguity problem in decryption without adding extra information for removing the ambiguity. Williams in [331] also proposed a M^3 *encryption* variant to Rabin but eliminated the 9 : 1 ciphertext ambiguity problem. The encryption is also proved to be as hard as factoring, although is is again still not the general IFP problem, since $N = pq$ was chosen to be

$$p \equiv q \equiv 1 \pmod 3$$

and

$$\frac{(p-1)(q-1)}{9} \equiv -1 \pmod 3.$$

LUC Cryptoystem

In 1993, Smith and Lennon proposed an RSA analog, called LUC [305], based on Lucas sequences. Let a, b be non-zero integers and $D = a^2 - 4b$. Consider the equation $x^2 - ax + b = 0$; its discriminant is $D = a^2 - 4b$, and α and β are the two roots:

$$\begin{aligned} \alpha &= \frac{a + \sqrt{D}}{2} \\ \beta &= \frac{a - \sqrt{D}}{2}. \end{aligned}$$

So

$$\begin{aligned} \alpha + \beta &= a \\ \alpha - \beta &= \sqrt{D} \\ \alpha\beta &= b. \end{aligned}$$

We define the sequences (U_k) and (V_k) by

$$\begin{aligned} U_k(a,b) &= \frac{\alpha^k - \beta^k}{\alpha - \beta} \\ V_k(a,b) &= \alpha^k + \beta^k. \end{aligned}$$

In particular, $U_0(a,b) = 0, U_1(a,b) = 1$, while $V_0(a,b) = 2, V_1(a,b) = a$. For $k \geq 2$, we also have

$$\begin{aligned} U_k(a,b) &= aU_{k-1} - bU_{k-2} \\ V_k(a,b) &= aV_{k-1} - bV_{k-2}. \end{aligned}$$

The sequences

$$\begin{aligned} U(a,b) &= (U_k(a,b))_{k\geq 0} \\ V(a,b) &= (V_k(a,b))_{k\geq 0} \end{aligned}$$

are called the *Lucas sequences* associated with the pair (a, b), in honour of the French mathematician François Edouard Lucas (1842–1891); more information about Lucas and Lucas sequences can be found e.g., in Yan [335]. The LUC cryptosystem works as follows (suppose Bob sends a ciphertext to Alice):

[1] **Key generation:** Alice publishes her public-key (e, N), satisfying

$$\begin{aligned} N = pq, &\quad \text{with} \quad p, q \in \text{ Primes}, \\ ed \equiv 1 \pmod{(p^2-1)(q^2-1)}, &\quad \text{with} \quad \gcd(e, (p^2-1)(q^2-1)) = 1. \end{aligned}$$

[2] **Encryption:** Bob encrypts his message $1 < M < N-1$ with $\gcd(M, N) = 1$ as follows:

$$C \equiv V_e(M, 1) \pmod{N}$$

based on the Lucas sequences, and sends it to Alice.

[3] **Decryption:** Alice performs the decryption as follows:

$$M \equiv v_d(C, 1) \pmod{N}.$$

[4] **Cryptoanalysis:** Anyone who can factor N can decrypt the message.

Theorem 2.5.6 (Correctness of LUC).

$$v_d(C, 1) \equiv M \pmod{N}.$$

Proof.

$$\begin{aligned} v_d(C,1) &\equiv v_d(V_e(M,1),1) \\ &\equiv v_{ed}(M,1) \\ &\equiv M \pmod{N}. \end{aligned}$$

□

Readers are suggested to consult [33], [205], [237] and [305] for more information about the LUC system, [33] also had a good discussion on both the LUC and the Dickson cryptosystems, as Dickson polynomials and the Lucas sequences are related to each other, and both can be used to construct RSA-type cryptosystems.

Elliptic Curve RSA

RSA has several noted elliptic curve analogues. Before introducing ing the EC analogue of RSA, we need two more results related the number of points on elliptic curves over the finite field $\mathbb{F}_p$.

Theorem 2.5.7 (Hasse). Let E be an elliptic curve over $\mathbb{F}_p$:

$$E_p(a, b): \ y^2 = x^3 + ax + b. \tag{2.36}$$

Then the number of points on $E_p(a, b)$, denoted by $\#(E_p(a, b))$, is as follows:

$$\#((E_p(a, b)) = 1 + p - \epsilon$$

where $|\epsilon| \le 2\sqrt{p}$.

Definition 2.5.1. Let $E_p(a, b)$ be an elliptic curve over the finite field $\mathbb{F}_p$. The complementary group of $E_p(a, b)$, denoted by $\overline{E_q(a, b)})$, is the set of points satisfying (2.36) together with a point at infinity $\mathcal{O}$, where y is of the form $u\sqrt{v}$, and v is a fixed non-quadratic residue modulo p and $v \in \mathbb{F}_p$.

Corollary 2.5.1. If
$$\#((E_p(a,b)) = 1 + p - \epsilon,$$
then
$$\#(\overline{E_q(a,b)})) = 1 + p + \epsilon.$$

The simplest EC analogue of RSA may be described as follows [335]:

[1] **Key generation:** $N = pq$ is the product of two large secret primes p and q. Choose two random integers a and b such that $E_N(a,b): \ y^2 = x^3 + ax + b$ defines an elliptic curve modulo both p and q. Let
$$N_n = \text{lcm}(\#(E_p(a,b)), \#(E_q(a,b))).$$
Choose a value for e such that
$$\begin{array}{rcl} \gcd(e, N_n) & = & 1, \\ ed & \equiv & 1 \pmod{N_n}. \end{array}$$
Publish (e, N) as public-key, but keep (d, p, q, N_n) as a secret.

[2] **Encryption:** To encrypt a message-point M, which is a point on $E_N(a,b)$, perform $C \equiv eM \pmod{N}$.

[3] **Decryption:** To encrypt a ciphertext C, just perform $M \equiv dC \pmod{N}$.

[4] **Cryptanalysis:** Anyone who can factor N, can of course decode the ciphertext.

In what follows, we introduce two relatively popular elliptic curve analogues of RSA. The first is the *KMOV cryptosystem* [180], which uses a family of supersingular elliptic curve $E_N(0,b): \ y^2 = x^3 + b$. An important property of this system is that if
$$p, q \equiv 2 \pmod{3},$$
then
$$N_n = \text{lcm}(p+1, q+1)$$
regardless of the value of b. The KMOV systems works as follows:

[1] **Key generation:** Let $N = pq$ be the product of two large secret primes p and q. Choose a supersingular elliptic curve
$$E_N(0,b): \ y^2 = x^3 + b$$
such that
$$N_n = \text{lcm}(p+1, q+1).$$
Choose a value for e such that
$$\begin{array}{rcl} \gcd(e, N_n) & = & 1, \\ ed & \equiv & 1 \pmod{N_n}. \end{array}$$
Publish (e, N) as public-key, but keep (d, p, q, N_n) as a secret.

[2] **Encryption:** To encrypt a message, e.g., $M = (m_1, m_2)$, Choose a suitable $b$such that

$$b \equiv m_2^3 - m_1^3 \pmod{N}$$

and C is computed by

$$C \equiv eM \pmod{N}$$

over $E_N(0, b)$.

[3] **Decryption:** To encrypt a ciphertext C, perform

$$M \equiv dC \pmod{N}$$

over $E_N(0, b)$.

[4] **Cryptanalysis:** Anyone who can factor N, can find the trap-door information (d, p, q, N_n), and hence can decode the message.

The second is the Demytko cryptosystem [98], which uses fixed (a, b) for elliptic curve

$$E_N(a, b) : \ y^2 = x^3 + ax + b,$$

In particular, it uses only the x-coordinate of the points of $E_p(a, b)$, The system relies on the fact if x is not the x-coordinate of a point on the elliptic curve $E_N(a, b)$, then it will be the x-coordinate of a point on of the *twisted* curve $\overline{E_p(a, b)}$. Thus,

$$N_n = \operatorname{lcm}(\#(E_p(a, b)), \#(\overline{E_p(a, b)}), \#(E_q(a, b)), \#(\overline{E_q(a, b)})).$$

The Demytko system works as follows:

[1] **Key generation:** Choose $N = pq$ with p and q primes. Choose an elliptic curve

$$E_N(0, b) : \ y^2 = x^3 + b$$

with fixed parameters p and q. Let

$$N_n = \operatorname{lcm}(\#(E_p(a, b)), \#(\overline{E_p(a, b)}), \#(E_q(a, b)), \#(\overline{E_q(a, b)})).$$

Choose a value for e such that

$$\begin{aligned} \gcd(e, N_n) &= 1, \\ ed &\equiv 1 \pmod{N_n}. \end{aligned}$$

Publish (e, N) as public-key, but keep (d, p, q, N_n) as a secret.

[2] **Encryption:** To encrypt a message M, compute

$$C \equiv eM \pmod{N}$$

over $E_N(a, b)$.

[3] **Decryption:** To encrypt a ciphertext C, perform

$$M \equiv dC \pmod{N}$$

over $E_N(a, b)$.

[4] **Cryptanalysis:** Anyone who can factor N, can recover M from C.

The cryptanalyst who knows the prime factorization of N, can decrypt the message easily. Readers are advised to consult [158] and [159] for some more recent developments in elliptic curve analogues of RSA.

ElGamal System

RSA is not only connected to the integer factorization problem, but also connected to the discrete logarithm problem, since, e.g., M can be found by taking the following discrete logarithm:

$$M \equiv \log_{M^e} M \pmod{N}$$

The first public-key cryptosystem based on discrete logarithms is the ElGamal cryptosystem (see Figure 2.11), proposed in 1985:

[1] A prime q and a generator $g \in \mathbb{F}_q^*$ are made public.

[2] Alice chooses a private integer $a = a_A \in \{1, 2, \cdots, q-1\}$. This a is the private decryption key. The public encryption key is $g^a \in \mathbb{F}_q$.

[3] Suppose now Bob wishes to send a message to Alice. He chooses a random number $b \in \{1, 2, \cdots, q-1\}$ and sends Alice the following pair of elements of $\mathbb{F}_q$:

$$(g^b, \; Mg^{ab})$$

where M is the message.

[4] Since Alice knows the private decryption key a, she can recover M from this pair by computing $g^{ab} \pmod{q}$ and dividing this result into the second element, i.e., Mg^{ab}.

[5] Someone who can solve the discrete logarithm problem in $\mathbb{F}_q$ breaks the cryptosystem by finding the secret decryption key a from the public encryption key g^a. In theory, there could be a way to use knowledge of g^a and g^b to find g^{ab} and hence break the cipher without solving the discrete logarithm problem. However, there is no known way to go from g^a and g^b to g^{ab} without essentially solving the discrete logarithm problem. So the security of the ElGamal cryptosystem is the same as the intractability of the discrete logarithm problem.

Surprisingly, the ElGamal cryptosystem (and in fact, almost all the existing systems, including RSA) can be easily extended to an elliptic curve cryptosystem. The following is the elliptic curve analog of the ElGamal cryptosystem (see also Figure 2.12).

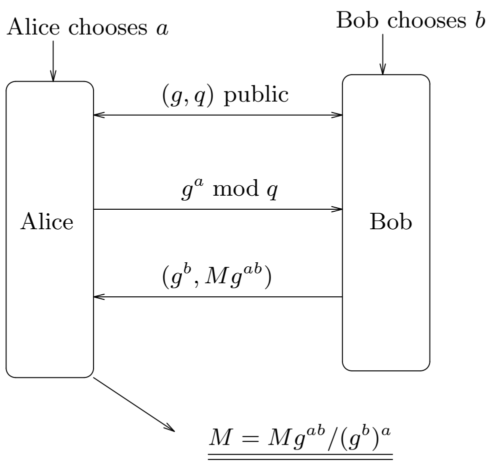


Figure 2.11. ElGamal System

[1] Alice and Bob publicly choose an elliptic curve E over $\mathbb{F}_q$ with $q = p^r$ and $p \in$ Primes, and a random *base* point $P \in E$.

[2] Alice chooses a random integer r_a and computes r_aP; Bob also chooses a random integer r_b and computes r_bP.

[3] To send a message-point M to Bob, Alice chooses a random integer k and sends the pair of points $(kP,\ M + k(r_bP))$.

[4] To read M, Bob computes

$$M + k(r_bP) - r_b(kP) = M. \tag{2.37}$$

[5] An eavesdropper who can solve the discrete logarithm problem on E can, of course, determine r_b from the publicly known information P and r_bP. Since there is no known efficient way to compute discrete logarithms, the system is secure.

Goldwasser-Micali System

The RSA encryption is *deterministic* in the sense that under a fixed public-key, a particular plaintext M is always encrypted to the same ciphertext C. Some of the drawbacks of such a deterministic scheme are:

(1) It is not secure for all probability distributions of the message space. For example, in RSA encryption, the messages 0 and 1 always get encrypted to themselves, and hence are easy to detect.

(2) It is easy to obtain some partial information of the secret key (p, q) from the public modulus n (assume that $n = pq$). For example, when the least-significant digit of n is 3, then it is easy to obtain the partial

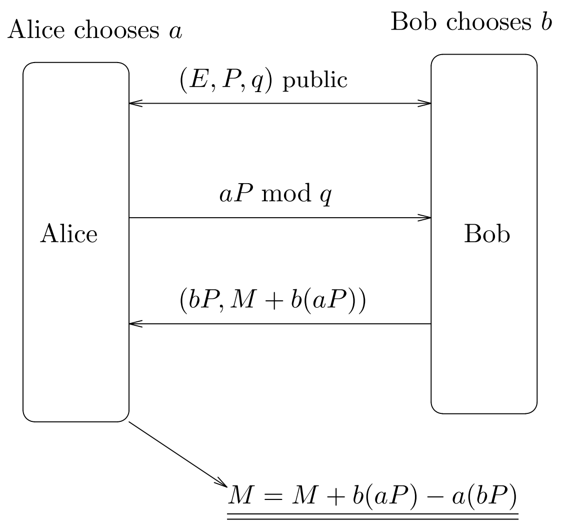


Figure 2.12. Elliptic Curve Analog of ElGamal System

information that the least-significant digits of p and q are either 1 and 3 or 7 and 9 since

$$\begin{array}{ll} 183 = 3 \cdot 61 & 253 = 11 \cdot 23 \\ 203 = 7 \cdot 29 & 303 = 3 \cdot 101 \\ 213 = 3 \cdot 71 & 323 = 17 \cdot 19. \end{array}$$

(3) It is sometimes easy to compute partial information about the plaintext M from the ciphertext C. For example, given (C, e, N), the Jacobi symbol of M over N can be easily deduced from C:

$$\left(\frac{C}{N}\right) = \left(\frac{M^e}{N}\right) \left(\frac{M}{N}\right)^e = \left(\frac{M}{N}\right).$$

d) It is easy to detect when the same message is sent twice.

Probabilistic encryption, or randomized encryption, however, utilizes randomness to attain a strong level of security, namely, the *polynomial security* and *semantic security*, defined as follows:

Definition 2.5.2. A public-key encryption scheme is said to be *polynomially* secure if no passive adversary can, in expected polynomial-time, select two plaintexts M_1 and M_2 and then correctly distinguish between encryptions of M_1 and M_2 with probability significantly greater that 1/2.

Definition 2.5.3. A public-key encryption scheme is said to be *semantically* secure if, for all probability distributions over the message space, whatever a passive adversary can compute in expected polynomial-time about the plaintext given the ciphertext, it can also be computed in expected polynomial time without the ciphertext.

Intuitively, a public-key encryption scheme is semantically secure if the ciphertext does not leak any partial information whatsoever about the plaintext that can be computed in expected polynomial-time. That is, given (C, e, N), it should be intractable to recover any information about M. Clearly, a public-key encryption scheme is semantically secure if and only if it is polynomially secure.

Recall that an integer a is a quadratic residue modulo N, denoted by $a \in Q_N$, if $\gcd(a, N) = 1$ and there exists a solution x to the congruence $x^2 \equiv a \pmod{N}$, otherwise a is a quadratic nonresidue modulo N, denoted by $a \in \overline{Q}_N$. The Quadratic Residuosity Problem (QRP) may be stated as follows:

Given positive integers a and n, decide whether or not $a \in Q_N$.

It is believed that solving QRP is equivalent to computing the prime factorization of N, so it is computationally infeasible. The Jacobi symbol $\left(\frac{x}{N}\right)$ is defined for any $x \in \mathbb{Z}_N$ and has a value in $\{1, -1\}$. If N is prime then

$$a \in Q_N \iff \left(\frac{a}{N}\right) = 1 \tag{2.38}$$

and if N is composite, then

$$a \in Q_N \implies \left(\frac{a}{N}\right) = 1 \tag{2.39}$$

but

$$a \in Q_N \overset{?}{\Longleftarrow} \left(\frac{a}{N}\right) = 1. \tag{2.40}$$

However

$$a \in \overline{Q}_N \Longleftarrow \left(\frac{a}{N}\right) = -1. \tag{2.41}$$

That is, whenever N is composite, a may belong to $\overline{Q}_N$ even if $\left(\frac{a}{N}\right) = 1$. Let $J_n = \{a \in (\mathbb{Z}/n\mathbb{Z})^* : \left(\frac{a}{N}\right) = 1\}$, then $\tilde{Q}_N = J_N - Q_N$. Thus, $\tilde{Q}_N$ is the set of all pseudosquares modulo N; it contains those elements of J_N that do not belong to Q_N.

In what follows, we present a cryptosystem whose security is based on the infeasibility of the Quadratic Residuosity Problem; it was first proposed by Goldwasser and Micali in 1984, under the term *probabilistic encryption.*

Algorithm 2.5.3 (Goldwasser-Micali Probabilistic Encryption). This algorithm uses the randomized method to encrypt messages and is based on the quadratic residuosity problem (QRP). The algorithm divides into three parts: key generation, message encryption and decryption.

[1] **Key generation**: Both Alice and Bob should do the following to generate their public and secret keys:
- Select two large distinct primes p and q, each with roughly the same size, say, each with β bits.
- Compute $N = pq$.
- Select a $y \in \mathbb{Z}_N$, such that $y \in \overline{Q}_N$ and $\left(\frac{y}{N}\right) = 1$. ($y$ is thus a pseudosquare modulo N).
- Make (N, y) public, but keep (p, q) secret.

[2] **Encryption**: To send a message to Alice, Bob should do the following:
- Obtain Alice's public-key (N, y).
- Represent the message m as a binary string $m = m_1 m_2 \cdots m_k$ of length k.
- For i from 1 to k do
  - Choose at random an $x \in (\mathbb{Z}_N)^*$ and call it x_i.
  - Compute c_i:

$$c_i = \begin{cases} x_i^2 \bmod N, & \text{if } m_i = 0, \quad \text{(r.s.)} \\ yx_i^2 \bmod N, & \text{if } m_i = 1, \quad \text{(r.p.s.)}, \end{cases} \tag{2.42}$$

    where r.s. and r.p.s. represent random square and random pseudosquare, respectively.
  - Send the k-tuple $c = (c_1, c_2, \cdots, c_k)$ to Alice. (Note first that each c_i is an integer with $1 \le c_i < N$. Note also that since n is a 2β-bit integer, it is clear that the ciphertext c is a much longer string than the original plaintext m.)

[3] **Decryption**: To decrypt Bob's message, Alice should do the following:
- For i from 1 to k do
  - Evaluate the Legendre symbol:

$$e_i' = \left(\frac{c_i}{p}\right)$$

  - Compute m_i:

$$m_i = \begin{cases} 0, & \text{if } \ e_i' = 1 \\ 1, & \text{if } \ \text{otherwise}. \end{cases} \tag{2.43}$$

    That is, $m_i = 0$ if $c_i \in Q_N$, otherwise, $m_i = 1$.
- Finally, get the decrypted message $m = m_1 m_2 \cdots m_k$.

One of the most important feature of the Goldwasser-Micali encryption is that

Theorem 2.5.8. The Goldwasser-Micali probabilistic encryption based on QRP is semantically secure.

Readers interested in semantically secure probabilistic encryption may wish to consult [124] and [121].

2.6 Chapter Notes and Further Readings

The underlying number-theoretic and computational complexity-theoretic ideas and concepts of the public-key cryptography in general and RSA cryptography in particular have been introduced in this chapter.

The idea of public-key cryptography was first publicly conceived and proposed by Diffie and Hellman [101], then both at Stanford University, in their seminal paper "New Directions in Cryptography" [101]; which was in turn based on some idea of Merkle [208], a PhD student of Hellman. Based on the work of Diffie, Hellman and Merkle (DHM), Rivest, Shamir and Addleman (RSA), then all at MIT, developed the first practical public-key cryptosystem in 1977 (see [115], [261], and [262]), now known as the RSA cryptosystem. These six people Diffie, Hellman, Merkle, Rivest, Shamir and Addleman are now regarded as the co-inventors of the public-key cryptography. Incidently, they jointly received the 1996 ACM Paris Kanellakis Theory and Practice Award "for the conception and first effective realization of public-key cryptography. The idea of a public-key cryptosystem was a major conceptual breakthrough that continues to stimulate research to this day, and without it today's rapid growth of electronic commerce would have been impossible." It is interesting to note that in December 1997 the Communication-Electronics Security Group (CESG) of the British Government Communications Headquarters (GCHQ), the successor of the Blechley Park (the British Government's Cryptography Centre during the 2nd World War for breaking the German Enigma codes), claimed that public-key cryptography was conceived (secretly) by James H. Ellis in 1970 and implemented by two of his colleagues Clifford C. Cocks and Malcolm J. Williamson between 1973 and 1976 in CESG, by declassifying five of their papers. The US Government's National Security Agency (NSA) also made a similar claim that they had public-key cryptography a decade earlier. However, according to the "first to publish, not first to keep secret" rule, the credit of the invention of public-key cryptography goes to Diffie, Hellman and Merkle for their seminar idea and to Rivest, Shamir and Adleman for their first implementation. The claims from CESG/GCHQ and NSA, however, are interesting footnote to the history of modern cryptography.

RSA is now the most popular cryptosystem used to safe-guard our private communications and/or business transactions over the insecure Internet. Readers are suggested to consult the following references for more information about RSA cryptography and cryptnanlysis: Buchmann [60], Delfs and Knebl [97], Wagstaff Jr. [321], Koblitz [172], [174], [175], Koblitz and Menezes [176], Konheim [179], Mollin [216], Mao [196], Menezes et al [207], Muller [225], Rothe [269], Salomann [274], Schneier [277], and Stamp and Low [310], Stinson [311], de Weger [323].

There are many variants of RSA, and there are many other types of cryptosystems whose security are also rely on the intractability of integer factorization problem or the related discrete logarithm problem. Notable systems on these directions include but are not limited to, Rabin's M^2 mod N system [252], Williams M^3 mod N system [327], ElGamal discrete logarithm system [106], Koblitz [170] and Miller [213] elliptic curve discrete logarithm systems, Goldwasser and Micali's quadratic residuosity system [124], and Goldwasser, Micali and Rackoff's zero-knowledge proof system [125].