数学图书 影印版系列

Neal Koblitz 著

密码学中的代数

Algebraic Aspects of Cryptography

清华大学出版社

数学图书 影 印 版 系 列

Neal Koblitz 著

密码学中的代数

Algebraic Aspects of Cryptography

清华大学出版社
北京

Neal Koblitz
Algebraic Aspects of Cryptography
ISBN: 3-540-63446-0

北京市版权局著作权合同登记号 图字：01-2006-6732

图书在版编目（CIP）数据

密码学中的代数 = Algebraic Aspects of Cryptography：英文 /（美）科比茨（Koblitz, N.）著. -- 北京：清华大学出版社, 2010.12
ISBN 978-7-302-24290-1

Ⅰ. ①密… Ⅱ. ①科… Ⅲ. ①代数—应用—密码—理论—英文 Ⅳ. ①TN918.1

中国版本图书馆 CIP 数据核字(2010)第 236208 号

责任编辑： 刘 颖
责任印制： 杨 艳

出版发行： 清华大学出版社 **地 址：** 北京清华大学学研大厦 A 座
http://www.tup.com.cn **邮 编：** 100084
社 总 机： 010-62770175 **邮 购：** 010-62786544
投稿与读者服务： 010-62776969, c-service@tup.tsinghua.edu.cn
质 量 反 馈： 010-62772015, zhiliang@tup.tsinghua.edu.cn
印 装 者： 北京国马印刷厂
发 行 者： 全国新华书店
开 本： 185×230 **印张：** 13.5
版 次： 2010 年 12 月第 1 版 **印 次：** 2010 年 12 月第 1 次印刷
印 数： 1~3000
定 价： 28.00 元

产品编号：015874-01

Preface

This book is intended as a text for a course on cryptography with emphasis on algebraic methods. It is written so as to be accessible to graduate or advanced undergraduate students, as well as to scientists in other fields. The first three chapters form a self-contained introduction to basic concepts and techniques. Here my approach is intuitive and informal. For example, the treatment of computational complexity in Chapter 2, while lacking formalistic rigor, emphasizes the aspects of the subject that are most important in cryptography.

Chapters 4–6 and the Appendix contain material that for the most part has not previously appeared in textbook form. A novel feature is the inclusion of three types of cryptography – "hidden monomial" systems, combinatorial–algebraic systems, and hyperelliptic systems – that are at an early stage of development. It is too soon to know which, if any, of these cryptosystems will ultimately be of practical use. But in the rapidly growing field of cryptography it is worthwhile to continually explore new one-way constructions coming from different areas of mathematics. Perhaps some of the readers will contribute to the research that still needs to be done.

This book is designed not as a comprehensive reference work, but rather as a selective textbook. The many exercises (with answers at the back of the book) make it suitable for use in a math or computer science course or in a program of independent study.

I wish to thank the participants in the Mathematical Sciences Research Institute's Summer Graduate Student Program in Algebraic Aspects of Cryptography (Berkeley, 16–27 June 1997) for giving me the opportunity to test-teach parts of the manuscript of this book and for finding errors and unclarities that needed fixing. I am especially grateful to Alfred Menezes for carefully reading the manuscript and making many valuable corrections and suggestions. Finally, I would like to thank Jacques Patarin for letting me report on his work (some of it not yet published) in Chapter 4; and Alfred Menezes, Yi-Hong Wu, and Robert Zuccherato for agreeing to let me include their elementary treatment of hyperelliptic curves as an Appendix.

Seattle, September 1997 *Neal Koblitz*

Contents

Chapter 1. Cryptography

Broadly speaking, the term *cryptography* refers to a wide range of security issues in the transmission and safeguarding of information. Most of the applications of algebra and number theory have arisen since 1976 as a result of the development of *public key* cryptography.

Except for a brief discussion of the history of *private key* cryptography (pre-1976), we shall devote most of this chapter to the (generally more interesting) questions that arise in the study of public key cryptosystems. After discussing the idea of public key cryptography and its importance, we next describe certain prototypical public key constructions.

§ 1. Early History

A cryptosystem for message transmission means a map from units of ordinary text called *plaintext message units* (each consisting of a letter or block of letters) to units of coded text called *ciphertext message units*. The idea of using arithmetic operations to construct such a map goes back at least to the Romans. In modern terminology, they used the operation of addition modulo N, where N is the number of letters in the alphabet, which we suppose has been put in one-to-one correspondence with $\mathbb{Z}/N\mathbb{Z}$. For example, if $N = 26$ (that is, messages are in the usual Latin alphabet, with no additional symbols for punctuation, numerals, capital letters, etc.), the Romans might encipher single letter message units according to the formula $C \equiv P+3 \pmod{26}$. This means that we replace each plaintext letter by the letter three positions farther down the alphabet (with the convention that $X \mapsto A$, $Y \mapsto B$, $Z \mapsto C$). It is not hard to see that the Roman system – or in fact any cryptosystem based on a permutation of single letter message units – is easy to break.

In the 16th century, the French cryptographer Vigenère invented a variant on the Roman system that is not quite so easy to break. He took a message unit to be a block of k letters – in modern terminology, a k-vector over $\mathbb{Z}/N\mathbb{Z}$. He then shifted each block by a "code word" of length k; in other words, his map from plaintext to ciphertext message units was translation of $(\mathbb{Z}/N\mathbb{Z})^k$ by a fixed vector.

Much later, Hill [1931] noted that the map from $(\mathbb{Z}/N\mathbb{Z})^k$ to $(\mathbb{Z}/N\mathbb{Z})^k$ given by an invertible matrix with entries in $\mathbb{Z}/N\mathbb{Z}$ would be more likely to be secure

than Vigenère's simple translation map. Here "secure" means that one cannot easily figure out the map knowing only the ciphertext. (The Vigenère cipher, on the other hand, can easily be broken if one has a long string of ciphertext, by analyzing the frequency of occurrence of the letters in each arithmetic progression with difference k. It should be noted that, even though the Hill system cannot be easily broken by frequency analysis, it is easy to break using linear algebra modulo N if you know or can guess a few plaintext/ciphertext pairs.)

For the most part, until about 20 years ago only rather elementary algebra and number theory were used in cryptography. A possible exception was the use of shift register sequences (see [Golomb 1982] and Chapter 6 and §9.2 of [Lidl and Niederreiter 1986]).

Perhaps the most sophisticated mathematical result in cryptography before the 1970's was the famous theorem of information theory [Shannon 1949] that said, roughly speaking, that the only way to obtain perfect secrecy is to use a *one-time pad*. (A "one-time pad" is a Vigenère cipher with period $k = \infty$.)

The first harbinger of a new type of cryptography seems to have been a passage in a book about time-sharing systems that was published in 1968 [Wilkes 1968, p. 91-92]. In it, the author describes a new *one-way cipher* used by R. M. Needham in order to make it possible for a computer to verify passwords without storing information that could be used by an intruder to impersonate a legitimate user.

> In Needham's system, when the user first sets his password, or whenever he changes it, it is immediately subjected to the enciphering process, and it is the enciphered form that is stored in the computer. Whenever the password is typed in response to a demand from the supervisor for the user's identity to be established, it is again enciphered and the result compared with the stored version. It would be of no immediate use to a would-be malefactor to obtain a copy of the list of enciphered passwords, since he would have to decipher them before he could use them. For this purpose, he would need access to a computer and even if full details of the enciphering algorithm were available, the deciphering process would take a long time.

In [Purdy 1974] the first detailed description of such a *one-way function* was published. The original passwords and their enciphered forms are regarded as integers modulo a large prime p, and the "one-way" map from $\mathbb{Z}/p\mathbb{Z}$ to $\mathbb{Z}/p\mathbb{Z}$ is given by a polynomial $f(x)$ which is not hard to evaluate by computer but which takes an unreasonably long time to invert. Purdy used $p = 2^{64} - 59$ and $f(x) = x^{2^{24}+17} + a_1x^{2^{24}+3} + a_2x^3 + a_3x^2 + a_4x + a_5$, where the coefficients a_i were arbitrary 19-digit integers.

§ 2. The Idea of Public Key Cryptography

Until the late 1970's, all cryptographic message transmission was by what can be called *private key*. This means that someone who has enough information to encrypt messages automatically has enough information to decipher messages as

well. As a result, any two users of the system who want to communicate secretly must have exchanged keys in a safe way, e.g., using a trusted courier.

The face of cryptography was radically altered when Diffie and Hellman invented an entirely new type of cryptography, called *public key* [Diffie and Hellman 1976]. At the heart of this concept is the idea of using a one-way function for encryption.

Definition 2.1. Speaking informally, we say that a one-to-one function $f : X \to Y$ is "one-way" if it is easy to compute $f(x)$ for any $x \in X$ but hard to compute $f^{-1}(y)$ for most randomly selected y in the range of f.*

The functions used for encryption belong to a special class of one-way functions that remain one-way only if some information (the "decryption key") is kept secret. Again using informal terminology, we can define a *public key encryption function* (also called a "trapdoor" function) as a map from plaintext message units to ciphertext message units that can be feasibly computed by anyone having the so-called "public" key but whose inverse function (which deciphers the ciphertext message units) cannot be computed in a reasonable amount of time without some additional information (the "private" key).

This means that everyone can send a message to a given user using the same enciphering key, which they simply look up in a public directory. There is no need for the sender to have made any secret arrangement with the recipient; indeed, the recipient need never have had any prior contact with the sender at all.

It was the invention of public key cryptography that led to a dramatic expansion of the role of algebra and number theory in cryptography. The reason is that this type of mathematics seems to provide the best source of one-way functions. Later we shall discuss the most important examples.

A curious historical question is why public key cryptography had to wait until 1976 to be invented. Nothing involved in the idea of public key cryptography or the early public key cryptosystems required the use of 20th century mathematics. The first public key cryptosystem to be used in the real world – the RSA system (see below) – uses number theory that was well understood by Euler. Why had it not occurred to Euler to invent RSA and offer it to the military advisers of Catherine the Great in gratitude for her generous support for the Russian Imperial Academy of Sciences, of which he was a member?

A possible reason for the late development of the concept of public key is that until the 1970's cryptography was used mainly for military and diplomatic purposes, for which private key cryptography was well suited. However, with the increased computerization of economic life, new needs for cryptography arose. To cite just one obvious example, when large sums of money are transferred electronically, one must be able to prevent white-collar thieves from stealing funds and

* In some situations one wants a one-way function to have a stronger property, namely, that it is hard to compute any partial information about $f^{-1}(y)$ (for instance, whether it is an odd or even number) for most randomly selected y.

nosy computer hackers (or business competitors) from monitoring what others are doing with their money. Another example of a relatively new use for cryptography is to protect the privacy of data (medical records, credit ratings, etc.). Unlike in the military or diplomatic situation – with rigid hierarchies, long-term lists of authorized users, and systems of couriers – in the applications to business transactions and data privacy one encounters a much larger and more fluid structure of cryptography users. Thus, perhaps public key cryptography was not invented earlier simply because there was no real need for it until quite recently.

Another reason why RSA was not likely to have been discovered in Euler's time is that in those days all computations had to be done by hand. To achieve an acceptable level of security using RSA, it would have been necessary to work with rather large integers, for which computations would have been cumbersome. So Euler would have had difficulty selling the merits of RSA to a committee of skeptical tsarist generals.

In practice, the great value of public key cryptography today is intimately connected with the proliferation of powerful computer technology.

2.1 Tasks for Public Key Cryptography

The most common purposes for which public key cryptography has been applied are:

(1) *confidential message transmission*;

(2) *authentication* (verification that the message was sent by the person claimed and that it hasn't been tampered with), often using *hash functions* (see §3.2) and *digital signatures* (see §3.3); *password* and *identification* systems (proving authorization to have access to data or a facility, or proving that you are who you claim to be); *non-repudiation* (guarding against people claiming not to have agreed to something that they really agreed to);

(3) *key exchange*, where two people using the open airwaves want to agree upon a secret key for use in some private key cryptosystem;

(4) *coin flip* (also called *bit commitment*); for example, two chess players in different cities want to determine by telephone (or e-mail) who plays white;

(5) *secret sharing*, where some secret information (such as the password to launch a missile) must be available to k subordinates working together but not to $k-1$ of them;

(6) *zero knowledge proof*, where you want to convince someone that you have successfully solved a number-theoretic or combinatorial problem (for example, you have found the square root of an integer modulo a large unfactored integer, or you have 3-colored a map) without conveying any knowledge whatsoever of what the solution is.

These tasks are performed through various types of *protocols*. The word "protocol" simply means an orderly procedure in which people send messages to one another.

In §§3–5 we shall describe several usable cryptosystems that perform one or more of the above tasks. We should caution the reader that the cryptosystems

described in this book are *primitives*. In cryptography the term "primitive" means a basic ingredient in a cryptosystem. In order to construct a practical system one generally has to modify and combine these primitives in a careful way so as to simultaneously achieve various objectives related to security and efficiency. For the most part we shall not deal with the practical issues that arise when one does this. The best general reference for such issues is the *Handbook of Applied Cryptography* [Menezes, van Oorschot, and Vanstone 1996].

2.2 Probabilistic Encryption

Most of the number theory based cryptosystems for message transmission are *deterministic*, in the sense that a given plaintext will always be encrypted into the same ciphertext by anyone. However, deterministic encryption has two disadvantages: (1) if an eavesdropper knows that the plaintext message belongs to a small set (for example, the message is either "yes" or "no"), then she can simply encrypt all possibilities in order to determine which is the supposedly secret message; and (2) it seems to be very difficult to *prove* anything about the security of a system if the encryption is deterministic. For these reasons, *probabilistic encryption* was introduced in [Goldwasser and Micali 1982, 1984]. We shall later (in Chapter 5 and §2.2 of Chapter 6) see examples of probabilistic encryption.

On the negative side, probabilistic encryption systems sometimes are vulnerable to so-called *adaptive chosen-ciphertext attack* (see Exercise 11 of §3 of Chapter 5 and Exercise 6 of §2 of Chapter 6).

We shall next discuss two particularly important examples of public key cryptosystems – RSA and Diffie–Hellman/DSA. Both are connected with fundamental questions in number theory – factoring integers and discrete logarithms, respectively. Although the systems can be modified to perform most or all of the six tasks listed above, we shall describe protocols for only a few of these tasks (message transmission in the case of RSA, and key exchange and digital signature in the case of Diffie–Hellman).

§ 3. The RSA Cryptosystem

3.1 Encryption

Suppose that we have a large number of users of our system, each of whom might want to send a secret message to any one of the other users. We shall assume that the message units m have been identified with integers in the range $0 \leq m < N$. For example, a message might be a block of k letters in the Latin alphabet, regarded as an integer to the base 26 with the letters of the alphabet as digits; in that case $N = 26^k$. In practice, in the RSA system N is a number of between about 200 and 600 decimal digits.

Each user A (traditionally named Alice) selects two extremely large primes p and q whose product n is greater than N. Alice keeps the individual primes secret, but she publishes the value of n in a directory under her name. She also chooses at random an exponent e which must have no common factor with $p-1$ or $q-1$ (and probably has the same order of magnitude as n), and publishes that value along with n in the directory. Thus, her *public key* is the pair (n, e).

Suppose that another user B (Bob) wants to send Alice a message m. He looks up her public key in the directory, computes the least nonnegative residue of m^e modulo n, and sends Alice this value (let c denote this *ciphertext* value). Bob can perform the modular exponentiation $c \equiv m^e \pmod{n}$ very rapidly (see Example 3.5 of Chapter 2).

To decipher the message, Alice uses her secret *deciphering key* d, which is any integer with the property that $de \equiv 1 \pmod{p-1}$ and $de \equiv 1 \pmod{q-1}$. She can find such a d easily by applying the extended Euclidean algorithm to the two numbers e and l.c.m.$(p-1, q-1)$ (see Example 3.4 of Chapter 2; here "l.c.m." means "least common multiple"). One checks (see Exercise 1 below) that if Alice computes the least nonnegative residue of c^d modulo n, the result will be the original message m.

What would prevent an unauthorized person C (Catherine) from using the public key (n, e) to decipher the message? The problem for Catherine is that without knowing the factors p and q of n there is apparently no way to find a deciphering exponent d that inverts the operation $m \mapsto m^e \pmod{n}$. Nor does there seem to be any way of inverting the encryption other than through a deciphering exponent. Here I use the words "apparently" and "seem" because these assertions have not been proved. Thus, one can only say that *apparently* breaking the RSA cryptosystem is as hard as factoring n.

3.2 Hash Functions

Before discussing digital signatures, it is necessary to explain what a hash function is. Suppose that we are sending a message containing l symbols, and we would like our signature to be much shorter – say, about k symbols. Here is an informal definition of "hash".

Definition 3.1. A function $H(x)$ from the set of l symbols to the set of k symbols is called a *hash function* if $H(x)$ is easy to compute for any x, but

1) no one can feasibly find two different values of x that give the same $H(x)$ ("collision resistant"); and
2) given y in the image of H, no one can feasibly find an x such that $H(x) = y$ ("preimage resistant").

Much research has been devoted to both the theory and practical implementation of hash functions. We shall not dwell on this. In practice it is not very hard to find a function that satisfies the properties in Definition 3.1.

One of the main uses of a hash function is in digital signatures. Suppose that Bob sends Alice a long message x of l symbols. Both Alice and Bob are using the

same hash function – and, in fact, there is no need for them to keep it secret from their adversary Catherine. After Bob sends Alice the message x, he appends the hash value $H(x)$. Alice would like to be certain that it was really Bob who sent the message x, and that Catherine did not alter his message before Alice received it. Suppose that she can somehow be certain that at least the appended $H(x)$ really did come from Bob. In that case all she has to do is apply the hash function to the message she received. If it agrees with $H(x)$, then she is happy: she knows that Catherine could not feasibly have tampered with x in such a way as to produce a distorted message x' such that $H(x') = H(x)$. The problem that remains is how Alice can be sure that $H(x)$ really came from Bob.

3.3 Signature

Here is how the last problem – how to be certain that $H(x)$ really came from Bob – can be solved using RSA. For convenience, choose k so that messages of length k are just small enough to make up one message unit; if the 26-letter Latin alphabet is being used, then k is the same as at the beginning of §3.1. After sending the message x, Bob computes the hash value $H = H(x)$. He does not simply send H to Alice, but rather first raises it to the power of his *deciphering exponent* d_{Bob} modulo n_{Bob}. Then Bob sends Alice the whole message x with $H' = H^{d_{\text{Bob}}} \pmod{n_{\text{Bob}}}$ appended, using Alice's enciphering exponent e_{Alice} and her modulus n_{Alice}. That is, he sends

$$\left(H^{d_{\text{Bob}}} \pmod{n_{\text{Bob}}}\right)^{e_{\text{Alice}}} \pmod{n_{\text{Alice}}} ,$$

where the notation $a \pmod{n}$ denotes the least nonnegative residue of a modulo n. After Alice deciphers the message, she takes the last message unit (which will look to her like gibberish rather than an intelligible plaintext message unit) and raises it to the power of Bob's *enciphering exponent* e_{Bob} modulo n_{Bob} in order to recover H. She then applies the hash function to the message, and verifies that the result coincides with H. Here the crucial observation is that Alice knows that *only Bob* would know the exponent that is inverted by raising to the e_{Bob}-th power modulo n_{Bob}. Thus, she knows that it really was Bob who sent her H. She also knows that it was he who sent the message x, which she received without any tampering.

It should be noted that this RSA signature has two other features besides simply allowing Alice to verify that it was in fact Bob who sent the message. In the first place, because the appended segment H' was encrypted along with the rest of the message, Bob's privacy is preserved; from the ciphertext an eavesdropper will not be able to find out who sent the message. In the second place, the signature ensures *non-repudiation*; that is, Bob cannot subsequently deny having sent the message.

§ 4. Diffie–Hellman and the Digital Signature Algorithm

The second landmark example of a public key cryptographic system is based on the discrete logarithm problem. First we define this problem.

Let $\mathbb{F}_p^* = (\mathbb{Z}/p\mathbb{Z})^* = \{1, 2, \ldots, p-1\}$ denote the multiplicative group of integers modulo a prime p. (This group will be treated in more detail in §2 of Chapter 3.) Let $g \in \mathbb{F}_p^*$ be a fixed element (our "base"). The *discrete log problem* in $\mathbb{F}_p^*$ to the base g is the problem, given $y \in \mathbb{F}_p^*$, of determining an integer x such that $y = g^x$ (if such x exists; otherwise, one must receive an output to the effect that y is not in the group generated by g).

4.1 Key Exchange

The Diffie–Hellman key exchange works as follows. Suppose that Alice and Bob want to agree upon a large integer to serve as a key for some private key cryptosystem. This must be done using open communication channels – that is, any eavesdropper (Catherine) knows everything that Alice sends to Bob and everything that Bob sends to Alice. Alice and Bob first agree on a prime p and a base element g in $\mathbb{F}_p^*$. This has been agreed upon publicly, so that Catherine also has this information at her disposal. Next, Alice secretly chooses a random positive integer $k_{\text{Alice}} < p$ (of about the same magnitude as p), computes the least positive residue modulo p of $g^{k_{\text{Alice}}}$ (see Example 3.5 of Chapter 2), and sends this to Bob. Meanwhile, Bob does likewise: he sends $g^{k_{\text{Bob}}} \in \mathbb{F}_p^*$ to Alice, while keeping k_{Bob} secret. The agreed upon key will then be the integer

$$g^{k_{\text{Alice}}k_{\text{Bob}}} \in \mathbb{F}_p^* = \{1, 2, \ldots, p-1\} ,$$

which Bob can compute by raising the integer he received from Alice to his secret k_{Bob}-power modulo p, and Alice can compute by raising the integer she received from Bob to the k_{Alice}-power modulo p. This works because in $\mathbb{F}_p^*$ we have

$$g^{k_{\text{Alice}}k_{\text{Bob}}} = \left(g^{k_{\text{Alice}}}\right)^{k_{\text{Bob}}} = \left(g^{k_{\text{Bob}}}\right)^{k_{\text{Alice}}} .$$

The problem facing the adversary Catherine is the so-called *Diffie–Hellman problem*: Given g, g^{k_A}, $g^{k_B} \in \mathbb{F}_p^*$, find $g^{k_A k_B}$. It is easy to see that anyone who can solve the discrete log problem in $\mathbb{F}_p^*$ can then immediately solve the Diffie–Hellman problem as well. The converse is not known. That is, it is conceivable (though thought to be unlikely) that someone could invent a way to solve the Diffie–Hellman problem without being able to find discrete logarithms. In other words, breaking the Diffie–Hellman key exchange has not been *proved* to be equivalent to solving the discrete log problem (although some recent partial results in this direction support the conjectured equivalence of the two problems; see [Boneh and Lipton 1996]). For practical purposes it is probably safe to assume that the Diffie–Hellman key exchange is secure provided that the discrete logarithm problem is intractable.

4.2 The Digital Signature Algorithm (DSA)

In 1991 the U.S. government's National Institute of Standards and Technology (NIST) proposed a Digital Signature Standard (DSS) based on a certain Digital Signature Algorithm (DSA). The role of DSS is expected to be analogous to that of the older Data Encryption Standard (DES): it is supposed to provide a standard digital signature method for use by government and commercial organizations. But while DES is a classical ("private key") cryptosystem, in order to construct digital signatures it is necessary to use public key cryptography. NIST chose to base their signature scheme on the discrete log problem in a prime finite field $\mathbb{F}_p$. The DSA is very similar to a signature scheme that was originally proposed in [Schnorr 1990]. It is also similar to a signature scheme in [ElGamal 1985a]. We now describe how the DSA works.

To set up the scheme (in order later to be able to sign messages), each user Alice proceeds as follows:

1) she chooses a prime q of about 160 bits (to do this, she uses a random number generator and a primality test);
2) she then chooses a second prime p that is $\equiv 1 \pmod{q}$ and has about 500 bits (more precisely, the recommended number of bits is a multiple of 64 between 512 and 1024);
3) she chooses a generator g of the unique cyclic subgroup of $\mathbb{F}_p^*$ of order q (she does this by computing $g_0^{(p-1)/q} \pmod{p}$ for a random integer g_0; if this number is not equal to 1, it will be a generator);
4) she takes a random integer x in the range $0 < x < q$ as her secret key, and sets her public key equal to $y = g^x \pmod{p}$.

Now suppose that Alice wants to sign a message. She first applies a hash function to her plaintext (see §3.2), obtaining an integer H in the range $0 < H < q$. She next picks a random integer k in the same range, computes $g^k \pmod{p}$, and sets r equal to the least nonnegative residue modulo q of the latter number (that is, g^k is first computed modulo p, and the result is then reduced modulo the smaller prime q). Finally, Alice finds an integer s such that $sk \equiv H + xr \pmod{q}$. Her signature is then the pair (r, s) of integers modulo q.

To verify the signature, the recipient Bob computes $u_1 = s^{-1}H \pmod{q}$ and $u_2 = s^{-1}r \pmod{q}$. He then computes $g^{u_1}y^{u_2} \pmod{p}$. If the result agrees modulo q with r, he is satisfied. (See Exercise 2 at the end of the chapter.)

This signature scheme has the advantage that signatures are fairly short, consisting of two numbers of 160 bits (the magnitude of q). On the other hand, the security of the system seems to depend upon intractability of the discrete log problem in the multiplicative group of the rather large field $\mathbb{F}_p$. Although to break the system it would suffice to find discrete logs in the smaller subgroup generated by g, in practice this seems to be no easier than finding arbitrary discrete logarithms in $\mathbb{F}_p^*$. Thus, the DSA seems to have attained a fairly high level of security without sacrificing small signature storage and implementation time.

There is a variant of DSA using elliptic curves that might be even harder to break than the finite-field DSA described above. This elliptic curve version will be discussed in Chapter 6.

§ 5. Secret Sharing, Coin Flipping, and Time Spent on Homework

5.1 Secret Sharing

Suppose that you want to give enough information to a group of people so that a secret password – which we think of as an integer N – can be determined by any group of k of them; but if only $k-1$ collaborate, they won't get anywhere. Here is a way to do this. Choose an arbitrary point $P = (x_1, \ldots, x_k)$ in the Euclidean space $\mathbb{R}^k$, where the x_i are integers and $x_1 = N$. Give each person in the group a single linear equation in k variables that is satisfied by P. Each equation determines a hyperplane in $\mathbb{R}^k$ that contains P. Choose your equations so that any k of them are linearly independent. (In other words, the coefficient matrix of any k of the equations has nonzero determinant.) Then any k people can solve the corresponding $k \times k$ system of linear equations for the point P. But $k-1$ equations determine a line, and so give no information about the first coordinate of P. (Here we're assuming that the line is not contained in the first coordinate hyperplane; a judicious choice of the linear equations will guarantee this.)

Another method of secret sharing is to choose a prime p for each person, and give him or her the value of the least nonnegative residue of N modulo p. N must be in a range where it can be uniquely recovered (using the Chinese Remainder Theorem, see Exercise 9 in §3 of Chapter 2) from its set of remainders modulo p for k values of p, but not from its remainders for $k-1$ values of p.

5.2 Bit Commitment

Suppose that Alice and Bob want to decide who gets a certain advantage – for example, who gets to play white in a chess match, or whose city gets to be the home team for the volleyball championship game. They can determine this by flipping a coin, provided that they are in the same physical location and both trust the fairness of the coin. Alternatively, they can "shoot fingers" – again, supposing that they are in the same place. That is, one of them (say, Alice) calls out "evens". Then they simultaneously throw out either one or two fingers. If the sum of the fingers is even (in other words, 2 or 4), then Alice wins. If the sum of the fingers is odd (in other words, 3), then Bob wins.

A cryptographic problem arises when Alice and Bob are far away from one another, and when they must act sequentially rather than at the same instant. In that case they need a procedure for *bit commitment*.