Number Theory and related Algorithms in Cryptography

By Alireza Nemaney Pour

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Hajime Ishihara

September, 2002

Number Theory and related Algorithms in Cryptography

By Alireza Nemaney Pour (010003)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Hajime Ishihara

and approved by
Associate Professor Hajime Ishihara
Professor Hiroakira Ono
Professor Atsushi Ohori

August, 2002 (Submitted)

Abstract

This research proposes a new Public Key Distribution Protocol for two goals: secure against an active adversary, and capable for key authentication. This protocol which is based on Diffie-Hellman Problem is a two-pass protocol and has many of desirable security. The protocol establishes a shared secret key K between two entities. The protocol is an extension of Diffie-Hellman Key Exchange using random numbers. This research also follows up MTI which is grounded in the two-pass key agreement. Generally, the research will focus on the developing of a protocol by which users can authenticate each other in an insecure network without central authority.

Contents

Chapter 1

Introduction

Cryptography has a long and fascinating history. The most striking development in the history of cryptography came in 1976 when **Diffie** and **Hellman** published "*New Directions in Cryptography* "[14]. This paper introduced the revolutionary concept of public-key cryptography and also provided a new and ingenious method for key exchange, the security of which is based on the intractability of the discrete logarithm problem and it is a commonly used protocol for key exchange. Later several public cryptosystems followed using many different underlying ideas. Many of them were soon proven to be insecure. However, the Diffie-Hellman protocol appears to has remained one of the strongest up to now.

Diffie-Hellman key exchange algorithm, is based on the assumption that discrete logarithms are hard to compute. This intractability hypothesis is also the foundation for the presumed security of a variety of other public key schemes. While there have been substantial advances in discrete log algorithms in the last two decades, in general the discrete log still appears to be hard. Unfortunately no proofs of hardness are available in this area, so it is necessary to rely on experience and intuition in judging what parameters to use for cryptosystems.

In many cryptographical protocols two parties wish to begin communicating. However, assume they do not initially possess any common secret key and thus cannot use secret key cryptosystems. The key exchange by Diffie-Hellman protocol remedies this situation by allowing the construction of a common secret key over an insecure communication channel. It is based on a problem related to discrete logarithms, namely the Diffie-Hellman problem. This problem is considered to be hard, and it is in some instances as hard as the discrete logarithm problem. The Diffie-Hellman protocol is generally considered to be secure when an appropriate mathematical group is used. In particular, the generator element used in the exponentiations should have a large period.

The objective of a key distribution or key agreement protocol is that, at the end of the protocol, the two parties involved both have possession of the same key K, and the value of K is not known to any other party. Certainly it is much more difficult to design a protocol providing this type of security.

Attacks against Diffie-Hellman include the *man-in-the-middle attack*. It is in practice very easy if the protocol doesn't use countermeasures such as authentication. Since

the network is insecure of n users, we need to protect against potential opponents (also called *adversary, intruder, enemy, attacker, eavesdropper*, and *impersonator* under various circumstances). An opponent might be a *passive adversary* who attempts to defeat a cryptographic technique by simply recording data and thereafter analyzing through a *passive attack*. On the other hand, an *active attack* involves an *active adversary* who modifies or injects messages. An active adversary can do various types of nasty things such as the following

1. alter messages that he observes being transmitted over the network.
2. save messages for reuse at a later time
3. attempt to masquerade as various users in the network

The objective of an active adversary might be one of the following :

1. to fool the users into accepting an "invalid " key as valid
2. to make the users believe that they have exchanged a key with each other when they have not.

1.1 Goals of the Research

In most of the Key-Exchange Algorithms a *trusted authority* is responsible for verifying the identities of users, choosing and transmitting keys to users, etc. This is something that this research follows to avoid it. Because there are cases that users want to communicate directly.

As a fundamental goal, the objective of this research is on developing a secure key distribution protocol aiming at achieving the following goals :

1. secure against the man-in-the-middle attack
2. security based on intractability of Diffie-Hellman problem
3. capable to key authentication, to provide assurance for the recipient whether he or she has computed the valid key

This research will focus on the development of a protocol by which users can authenticate each other in an insecure network without a central authority. Using such a protocol, users will be able to correctly identify the origin of a message, with an assurance that the identity is not false.

1.2 Related Works

There are many algorithms based on the intractablility of the Discrete Logarithm Problem which most of them are secure against a passive attack, but insecure against the man-in-the-middle attack which some of them will be discussed in this research.

Yamamoto and Akiyama proposed a protocol called Method 1 of key agreement protocol[8, p178] which is an extension to Diffie-Hellman protocol using random numbers for session keys. This work is secure against a passive attack but it has problems with the man-in-the-middle attack.

There is another work by Okamoto and Nakamura proposing Method 2 [8, p179] of key agreement protocol using random numbers for session keys in a different way than Method 1. The security of this work is almost the same as Method 1.

Matsumoto, Takashima and Imai have constructed several interesting key agreement protocols by modifying Diffie-Hellman key Exchange. These protocols are classified to **MTI**[19] key agreement protocols. We present some of these protocols and consider the man-in-middle attack about one of them to show how an active adversary can fool the users into accepting an "invalid " key as valid.

Elgamal protocol is another work which is much different from the methods mentioned above. This one-pass protocol gives authentication but it is not secure against the man-in-the-middle attack.

Authentication is a second problem with these protocls. They are unauthenticated protocols which users cannot assure whether they have generated the correct key. In some of the protocols like ElGamal or one-pass protocols the recipient has no corroboration of whom it shared the secret key with, nor any key freshness assurances. Neither party obtains entity authentication or key confirmation.

1.3 Outline of the Research

This research is organized into 5 chapters. Chapter 1 gives an intoduction to historical background of Diffie-Hellman Key Exchange as well as related works. Chapter 2 presents a mathematical background of the basics needed to understand the Diffie-Hellman based on protocols and the types of attacks it is vulnerable to. Diffie-Hellman based on protocols require knowledge of several areas of mathematics, including number theory, groups, rings and fields. It has been tried to focus on the most important and neccessary definitions and theorems which are needed to understand this research. Of course, no proof about the theorems has been included, further background and of the theorems can be found in given references.

In Chapter 3 Diffie-Hellman, Elgamal, and MTI protocols will be discussed. The objective of this chapter is to review Diffie-Hellman based on protocols needed to understand the next chapter. Moreover, the man-in-the-middle attack will be considered on Diffie-Hellman protocol and MTI/A0.

Chapter 4 which is the main part of this paper, consists of the protocol which is proposed in this research. This protocol is extended from Diffie-Hellman and MTI/C1 protocols.

The security points will be discussed in the rest of this chapter. Two cases of the man-in-the-middle attack have been considered about this protocol. Most of these attacks are based on mathematical tricks that it has been tried to consider the most important ones.

The last chapter consists of the conclusions and future works which the results and problems will be considered.

Chapter 2

Mathematical Background

This chapter is a collection of basic material on Number Theory, Groups, Rings, Fields, and Finit Fields that will be used throughout this research. The purpose of the chapter is to recall the notation and facts from Number Theory which we will need to have at our fingertips in our later work. Further background and proofs of the facts presented here can be found in references given in [1, 2, 4, 8, 9, 10, 12, 13].

2.1 Topics in Number Theory

2.1.1 Divisibility

The set of integers $\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$ is denoted by symbol $\mathbb{Z}$.

Definition 2.1.1 Let a, b be integers. Then a *divides* b (equivalently: a is a *divisor* of b, or a is a *factor* of b) if there exists an integer c such that $b = ac$. if a divides b, then this is denoted by $a|b$.

Example 2.1.1 (i) $-3|18$, since $18 = (-3)(-6)$. (ii) $173|0$, since $0 = (173)(0)$.

The following are some elementary properties of divisibility.

Proposition 2.1.1 (*properties of divisibility*)

1. $a|a$.
2. If $a|b$ and $b|c$, then $a|c$.
3. If $a|b$ and $a|c$ then $a|(bx + cy)$ for all $x, y \in \mathbb{Z}$.
4. If $a|b$ and $b|a$, then $a = \pm b$.

Definition 2.1.2 (*division algorithm for integers*) if a and b are integers with $b \geq 1$, then ordinary long division of a by b yields integers q (the *quotient*) and r (the *remainder*) such that

$$a = qb + r, \text{ where } 0 \leq r < b.$$

Moreover, q and r are unique. The remainder of the division is denoted $a \ mod \ b$, and the quotient is denoted $a \ div \ b$.

Example 2.1.2 if $a = 73$, $b = 17$, then $q = 4$ and $r = 5$. Hence $73 \ mod \ 17 = 5$ and $73 \ div 17 = 4$.

Definition 2.1.3 An integer c is a *common divisor* of a and b if $c|a$ and $c|b$.

Definition 2.1.4 A non-negative integer d is the *greatest common divisor* of integers a and b, denoted $d = gcd(a, b)$, if

1. d is a common divisor of a and b; and
2. whenever $c|a$, and $c|b$, then $c|d$.

Equivalently, $gcd(a, b)$ is the largest positive integer that divides both a and b, with the exception that $gcd(0, 0) = 0$.

Example 2.1.3 The common divisors of 12 and 18 are $\{\pm 1, \pm 2, \pm 3, \pm 6\}$, and $gcd(12, 18) = 6$.

Definition 2.1.5 A non-negative integer d is the *least common multiple* of integers a and b, denoted $l = lcm(a, b)$, if

1. $a|l$ and $b|l$; and
2. whenever $a|c$, and $b|c$, then $l|c$.

Equivalently, $lcm(a, b)$ is the smallest non-negative integers divisible by both a and b.

Theorem 2.1.1 If a and b are positive integers, then $lcm(a, b) = a \cdot b/gcd(a, b)$.

Example 2.1.4 Since $gcd(12, 18) = 16$, it follows that $lcm(12, 18) = 12 \cdot 18/6 = 36$.

Definition 2.1.6 (*prime numbers*) An $p \geq 2$ is said to be *prime* if its only positive divisors are 1 and p. Otherwise, p is called *composite*.

Theorem 2.1.2 If p is prime and $p|ab$, then either $p|a$ or $p|b$(or both).

Theorem 2.1.3 There are an infinite number of prime numbers.

Proof.
Let us assume that there are only finitely many primes, then we can list them all:

$$p_1, p_2, \cdots, p_r.$$

Let P be their product, a very big number but still finite:

$$P = p_1 \times p_2 \times \cdots \times p_r.$$

We now consider $P + 1$ which is an integer and so can be factored into primes. But since all of the primes divide P, none of them divide $P + 1$, since if p_i divides P and it divides $P + 1$, then it must divide 1. This our contradiction.

Observe that all that this proof does for us is prove that there are infinitely many primes. It is useless in trying to generate the primes. If we know the first n primes, this will give us a new prime, but probably not the next prime. Also, this does not promise that $P + 1$ will be a prime. For example:

$$(2 \times 3 \times 5 \times 7 \times 11 \times 13) + 1 = 30031 = 59 \times 509.$$

Theorem 2.1.4 (*fundamental theorem of arithmetic*) Factorization into primes is unique up to order.

Proof.
We will actually prove that every integer with non-unique factorization has a proper divisor with non-unique factorization. If there were integers with non-unique factorization, then eventually we would be reduced to a prime with non-unique factorization, and that would conradict the fact that it is a prime and thus has no positive divisors other than 1 and itself.

Let n be an integer with non-unique factorization:

$$\begin{aligned} n &= p_1 \times p_2 \times \cdots \times p_r \\ &= q_1 \times q_2 \times \cdots \times q_s, \end{aligned}$$

where the primes are not necessarily distinct, but where the second factorization is not simply a reordering of the first. The prime q_1 divides n and so it divides the product of the p_i's. By repeating this, there is at least one p_i which is divisible by q_1. If necessary, reorder the p_i's so that q_1 divides p_1. Since p_1 is prime, q_1 must equal p_1. This says that

$$\begin{aligned} \frac{n}{q_1} &= p_2 \times p_3 \times \cdots \times p_r \\ &= q_2 \times q_3 \times \cdots \times q_s. \end{aligned}$$

Since the factorization of n were distinct, there factorizations of n/q_1 must also be distinct. Therefore n/q_1 is proper divisor of n with non-unique factorization.

where the p_i are distinct primes, and the e_i are positive integers. Furthermore, the factorization is unique up to rearrangement of factors.

Theorem 2.1.5 If $a = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, $b = p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k}$, where each $e_i \geq 0$ and $f_i \geq 0$, then

$$gcd(a, b) = p_1^{min(e_1, f_1)} p_2^{min(e_2, f_2)} \cdots p_k^{min(e_k, f_k)}$$

and

$$lcm(a,b) = p_1^{max(e_1,f_1)} p_2^{max(e_2,f_2)} \cdots p_k^{max(e_k,f_k)}.$$

Example 2.1.5 Let $a = 4864 = 2^8 \cdot 19, b = 4358 = 2 \cdot 7 \cdot 13 \cdot 19$. Then $gcd(4864, 3458) = 2 \cdot 19 = 38$ and $lcm(4864, 3458) = 2^8 \cdot 7 \cdot 13 \cdot 19 = 442624$.

Definition 2.1.7 For $n \geq 1$, let $\varphi(n)$ denote the number of integers in the interval $[1, n]$ which are relatively prime to n. The function φ is called the *Euler phi function*.

Theorem 2.1.6 (*properties of Euler phi function*)

1. If p is a prime, then $\varphi(p) = p - 1$.
2. The Euler phi function is *multiplicative*. That is, if $gcd(m, n) = 1$, then $\varphi(mn) = \varphi(m) \cdot \varphi(n)$.
3. If $p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ is the prime factorization of n, then

$$\begin{aligned} \varphi(n) &= p_1^{e_1-1}(p_1 - 1) \times p_2^{e_2-1}(p_2 - 1) \times \cdots \times p_k^{e_k-1}(p_k - 1) \\ &= n\Big(1 - \frac{1}{p_1}\Big)\Big(1 - \frac{1}{p_2}\Big) \cdots \Big(1 - \frac{1}{p_k}\Big). \end{aligned}$$

2.1.2 Euclidean Algorithms

Let a and b be non-negative integers, each less than or equal to n. The number of bits in binary representation of n is $\lfloor \ln n \rfloor + 1$, and this number is approximated by $\ln n$. The number of bit operations for the four basic integer operations of addition , subtraction, multiplication, and division using classical algorithms is summarized in Table 2.1. Note

Operation		Bit complexity		
Addition	$a + b$	$O(\ln a + \ln b)$	$=$	$O(\ln n)$
Subtraction	$a - b$	$O(\ln a - \ln b)$	$=$	$O(\ln n)$
Multiplication	$a \cdot b$	$O((\ln a)(\ln b))$	$=$	$O((\ln n)^2)$
Division	$a = qb + r$	$O((\ln q)(\ln b))$	$=$	$O((\ln n)^2)$

Table 2.1: *Bit complexity of basic operations in* $\mathbb{Z}$

that, the above equations about the bit operations are well-known statements which furture proofs and informaition can be found in references given in [1,2]. The other way to write the above equations is as following. The notation $Time(A)$ denotes the number of bit operations for the job needed in A.

$Time(a + b) = O(\log(max(a, b)))$, bit operations, where $a, b \in \mathbb{Z}$.
$Time(a \times b) = O(\log a \log b)$, bit operations, where $a, b \in \mathbb{Z}$.

$Time(a/b) = O(\log a \log b)$, bit operations, where $a, b \in \mathbb{Z}$.
$Time(\sqrt{a}) = O(\log^3 a)$, bit operations, where $a \in \mathbb{Z}$.
$Time(g^a \ mod \ b) = O(\log a \log^2 b)$, where $a, b \in \mathbb{Z}$, for some fixed integer g.

For computing the greatest common divisor of two integers the most efficient algorithm is the Euclidean Algorithm which is based on the following simple fact.

Theorem 2.1.7 If a and b are positive integers with $a > b$, then $gcd(a, b) = gcd(b, a \ mod \ b)$.

The Euclidean algorithm consists of performing the following sequence of divisions Then

$$\begin{array}{rll} a & = q_1 b + r_1, & 0 < r_1 < b \\ b & = q_2 r_1 + r_2, & 0 < r_2 < r_1 \\ r_1 & = q_3 r_2 + r_3, & 0 < r_3 < r_2 \\ \vdots & & \\ r_{n-2} & = q_n r_{n-1} + r_n, & 0 < r_n < r_{n-1} \\ r_{n-1} & = q_{n+1} r_n + 0, & \end{array}$$

$$(a > b > r_1 > r_2 > \cdots > r_n > 0)$$

the greatest common divisor will be

$$gcd(a, b) = gcd(b, r_1) = gcd(r_2, r_1) = \cdots = gcd(r_n, 0) = r_n.$$

Hence, it follows that $gcd(a, b) = r_n$. Further information can be found in references [9,12].

Algorithm Euclidean algorithm for computing the greatest common divisor of two integers
INPUT: two non-negative integers a and b with $a \geq b$.
OUTPUT: the greatest common divisor of a and b.
1. While $b \neq 0$ do the following:
1.1 Set $r \leftarrow a \ mod \ b, a \leftarrow b, b \leftarrow r$.
2. Return(a).

Table 2.2: *Euclidean Algorithm*

Theorem 2.1.8 The above algorithm has a running time of $O((\ln n)^2)$ bit operations.

The Euclidean algorithm can be extended so that it not only yields the greatest common divisor d of two integers a and b, but also integers x and y satisfying $ax + by = d$.

Algorithm Extende Euclidean algorithm
INPUT: two non-negative integers a and b with $a \geq b$.
OUTPUT: $d = gcd(a, b)$ and integers x, y satisfying $ax + by = d$.
1. If $b = 0$ then set $d \leftarrow a$, $x \neq 1$, $y \leftarrow 0$ and return(d, x, y).
2. Set $x_2 \leftarrow 1, x_1 \leftarrow 0, y_2 \leftarrow 0, y_1 \leftarrow 1$.
3. While $b > 0$ do the following :
3.1 $q \leftarrow \lfloor a/b \rfloor, r \leftarrow a - qb, x \leftarrow x_2 - qx_1, y \leftarrow y_2 - qy_1$.
3.1 $a \leftarrow b, b \leftarrow r, x_2 \leftarrow x_1, x_1 \leftarrow x, y_2 \leftarrow y_1, andy_1 \leftarrow y$.
4. Set$d \leftarrow a, x \leftarrow x_2, y \leftarrow y_2$, and return$(d, x, y)$.

Table 2.3: *Extended Euclidean Algorithm*

Theorem 2.1.9 Extended Euclidean algorithm has running time of $O((\ln n)^2)$ bits operations.

Since the Euclidean algorithm computes the greatest common divisors, it can be used to determine if a positive integer $a < n$ has a multiplicative inverse(it will discussed in section 2) molulo n. However it does not compute the value of the multiplicative inverse.

2.1.3 Congruences

Let n be a positive integer.

Definition 2.1.8 If a and b are integers, then a is said to be *congruence to b modulo n*, written $a \equiv b \ (mod\ n)$, if n devides $(a - b)$. The integer n is called the *modulus* of the congruence.

Theorem 2.1.10 (*properties of congruences*) For all $a, a_1, b, b_1, c \in \mathbb{Z}$, the following are true.

1. $a \equiv b \ (mod\ n)$ if and only if a and b leave the same remainder when divided by n.
2. (*reflexivity*) $a \equiv a \ (mod\ n)$.
3. (*symmetry*) If $a \equiv b \ (mod\ n)$, then $b \equiv a \ (mod\ n)$.
4. (*transitivity*) If $a \equiv b \ (mod\ n)$, and $b \equiv c \ (mod\ n)$, then $a \equiv c \ (mod\ n)$.
5. If $a \equiv a_1 \ (mod\ n)$, and $b \equiv b_1 \ (mod\ n)$, then $a + b \equiv a_1 + b_1 \ (mod\ n)$ and $ab \equiv a_1 b_1 \ (mod\ n)$.

Definition 2.1.9 The *integers modulo n*, denoted $\mathbb{Z}_n$, is the set of (*equivalence classes of* integers) $\{0, 1, 2, \ldots, n - 1\}$. Addition, subtraction, and multiplication in $\mathbb{Z}_n$ are performed modulo n.

Example 2.1.6 $\mathbb{Z}_{25} = \{0, 1, 2, \ldots, 24\}$. In $\mathbb{Z}_{25}$, $13 + 16 = 4$, since $13 + 16 = 29 \equiv 4 \ (mod\ 25)$. Similarly, $13 \cdot 16 = 8$ in $\mathbb{Z}_{25}$.

Theorem 2.1.11 (*Chinese remainder theorem, CRT*) If the integers $n_1, n_2, \ldots, n_k$ are pairwise relatively prime, then the system of simultaneous congruences

$$\begin{array}{rcll} x & \equiv & a_1 & (mod\ n_1) \\ x & \equiv & a_2 & (mod\ n_2) \\ & \vdots & & \\ x & \equiv & a_k & (mod\ n_k) \end{array}$$

has a unique solution modulo $n = n_1 n_2 \cdots n_k$.

Example 2.1.7 The pair of congruences $x \equiv 3 \ (mod\ 7), x \equiv 7 \ (mod\ 13)$ has a unique solution $x \equiv 59 \ (mod\ 91)$.

Theorem 2.1.12 If $gcd(n_1, n_2) = 1$, then the pair of congruences $x \equiv a \ (mod\ n_1), x \equiv a \ (mod\ n_2)$ has a unique solution $x \equiv a \ (mod\ n_1 n_2)$.

Definition 2.1.10 The *multiplicative group* of $\mathbb{Z}_n$ is $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | gcd(a, n) = 1\}$. In particular, if n is a prime, then $\mathbb{Z}_n^* = \{a | 1 \leq a \leq n - 1\}$.

Definition 2.1.11 The *order* of $\mathbb{Z}_n^*$ is defined to be the number of elements in $\mathbb{Z}_n^*$, namely $|\mathbb{Z}_n^*|$.

Theorem 2.1.13 Let $n \geq 2$ be an integer.

1. (*Euler's Theorem*) If $a \in \mathbb{Z}_n^*$, then $a^{\varphi(n)} \equiv 1 \ (mod\ n)$.
2. If n is a product of distinct primes, and if $r \equiv s \ (mod\ \varphi(n))$, then $a^r \equiv a^s \ (mod\ n)$ for all integers a. In other words, when working modulo such an n exponents can be reduced modulo $\varphi(n)$.

Definition 2.1.12 Let $a, b \in \mathbb{Z}_n$. The *multiplicative inverse of* a modulo n is an integer $x \in \mathbb{Z}_n$ such that $ax \equiv 1 \ (mod\ n)$. If such an x exists, then it is unique, and a is said to be *invertible*, or a *unit*; the inverse of a is denoted by a^{-1}.

Theorem 2.1.14 Let $a \in \mathbb{Z}_n$. Then a is invertible if and only if $gcd(a, n) = 1$.

Proof.
First, if $gcd(a, n)$ were greater than 1, we could not have $ab = 1 \ (mod\ n)$ for any b, because that would imply that d divides $ab - 1$ and hence divides 1.

A special case of Euler's theorem is Fermat's (little) theorem.

Theorem 2.1.15 Let p be a prime.

1. (*Fermat's Theorem*) If $gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \ (mod\ p)$.

2. In particular, $a^p \equiv a \ (mod\ p)$ for all integers a.

Definition 2.1.13 Let $a \in \mathbb{Z}_n^*$. The *order* of a, is denoted ord(a), is the least positive integer t such that $a^t \equiv 1 \ (mod\ n)$.

Theorem 2.1.16 If the order of $a \in \mathbb{Z}_n^*$ is t, and $a^s \equiv 1 \ (mod\ n)$, then t divides s. In particular, $t|\varphi(n)$.

Definition 2.1.14 Let $a \in \mathbb{Z}_n^*$. If the order of g is $\varphi(n)$, then g is said to be a *generator* or a *primitive element* of $\mathbb{Z}_n^*$. If $\mathbb{Z}_n^*$ has a generator, then $\mathbb{Z}_n^*$ is said to be *cyclic*.

Theorem 2.1.17 (*properties of generators of* $\mathbb{Z}_n^*$)

1. $\mathbb{Z}_n^*$ has a generator if and only if $n = 2, 4, p^k$ or $2p^k$, where p is an odd prime and $k \geq 1$. In particular, if p is a prime, then $\mathbb{Z}_n^*$ has a generator.

2. If g is a generator of $\mathbb{Z}_n^*$, then $\mathbb{Z}_n^* = \{a^i \ (mod\ n) |\ 0 \leq i \leq \varphi(n) - 1\}$.

3. Suppose that g is a generator of $\mathbb{Z}_n^*$. Then $b = g^i \ (mod\ n)$ is also a generator of $\mathbb{Z}_n^*$ if and only if $gcd(i, \varphi(n)) = 1$. It follows that if $\mathbb{Z}_n^*$ is cyclic, then the number of generators is $\varphi(\varphi(n))$.

4. $g \in \mathbb{Z}_n^*$ is a generator of $\mathbb{Z}_n^*$ if and only if $g^{\varphi(n)/p} \not\equiv 1 \ (mod\ n)$ for each prime divisor p of $\varphi(n)$.

Definition 2.1.15 Let $a \in \mathbb{Z}_n^*$. a is said to be *quadratic residue* modulo n, or *square* modulo n, if there exists an $x \in \mathbb{Z}_n^*$ such that $x^2 \equiv a \ (mod\ n)$. If no such x exists, then a is called a *quadratic non-residue* modulo n. The set of all quadratic residues modulo n is denoted by Q_n and the set of all quadratic non-residues is denoted by $\bar{Q}_n$.

Example 2.1.8 $g = 6$ is a generator of $\mathbb{Z}_{13}^*$. The powers of g are listed in the following table. $Q_{13} = \{1, 3, 4, 9, 12\}$ and $\bar{Q}_{13} = \{2, 5, 6, 7, 8, 11\}$.

i	0	1	2	3	4	5	6	7	8	9	10	11
$a^i \ mod\ 13$	1	6	10	8	9	2	12	7	3	5	4	11

2.1.4 Repeated Squaring Method

A repeated squaring method is a basic computation in modular arithmetic for finding $b^n \ (mod\ m)$ when both m and n are very large. There is a clever way of doing this that is much quicker than repeated multiplication of b itself. The related algorithm is as following:

Let $n_0, n_1, \ldots, n_{k-1}$ denote the binary digits of n, i.e.,

$$n = n_0 + 2n_1 + 4n_2 + \cdots + 2^{k-1}n_{k-1} \qquad (n_j = 0 \ or 1).$$

Then, We compute as following :

$$b^2, (b^2)^2 = b^4, (b^4)^2 = b^8, \cdots, (b^{2^{k-2}})^2 = b^{2^{k-1}}.$$

It follows as

$$\begin{aligned} b^n &= b^{n_0+2n_1+4n_2+\cdots+2^{k-1}n_{k-1}} \\ &= b^{n_0} \cdot b^{2n_1} \cdot b^{4n_2} \ldots b^{2^{(k-1)}n_{k-1}} \\ &= b^{n_0} \cdot (b^2)^{n_1} \cdot (b^4)^{n_2} \cdots (b^{2^{k-1}})^{n_{k-1}} \end{aligned}$$

As $b^2, (b^2)^2 = b^4, (b^4)^2 = b^8, \cdots, (b^{2^{k-2}})^2 = b^{2^{k-1}}$ is computed before, $b^n \ (mod\ m)$ can be computed easily.

Example 2.1.9 Suppose we want to compute $432^{678} \ (mod\ 987)$. The basic trick is to start with a number and keep squaring:

$$432^2 = 186624 \equiv 81 \quad 432^4 \equiv 81^2 \equiv 639 \quad 432^8 \equiv 639^2 \equiv 690 \ldots 432^{512} \equiv 858$$

Since $678 = 512 + 128 + 32 + 4 + 2$,

$$432^{678} \equiv (81)(639) \ldots (858) \equiv 204 \qquad \text{(I hope!)}$$

Calculations with exponents involve not-too-many multiplications. If the numbers have several hundred digits, however, it is necessary to design special subroutines to do the multiplications.

The idea behind fast exponentiation is that if the exponent is a power of 2 then we can exponentiate by successively squaring:

$$\begin{aligned} x^8 &= ((x^2)^2)^2, \\ x^{256} &= (((((((x^2)^2)^2)^2)^2)^2)^2)^2. \end{aligned}$$

If the exponent is not a power of 2, then we use its binary representation, which is just a sum powers of 2:

$$x^{291} = x^{256} \times x^{32} \times x^2 \times x^1.$$

Thus to raise x to power n requires only about $\log n$ operations.

2.1.5 The Legendre and Jacobi symbols

The Legendre symbol is a useful tool for keeping track of whether or not an integer a is a quadratic residure modulo a prime p. Further background and proofs of the facts presented here can be found in references given in [10, 12, 13].

Definition 2.1.16 Let p be an odd prime and a an integer. The *Legendre Symbol* $\left(\frac{a}{p}\right)$ is defined to be

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{if } p|a, \\ 1, & \text{if } a \in Q_p, \\ -1, & \text{if } a \in \bar{Q}_p. \end{cases}$$

Theorem 2.1.18 (*properties of Legendre symbol*) Let p be a prime and $a, b \in \mathbb{Z}$. Then the Legendre symbol has the following properties:

1. $\left(\frac{a}{p}\right) \equiv a^{(p-1)/2}$ $(mod\ p)$. In paticular, $\left(\frac{1}{p}\right) = 1$ and $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$. Hence $-1 \in Q_p$ if $p \equiv 1$ $(mod\ 4)$, and $-1 \in \bar{Q}_p$ if $p \equiv 3$ $(mod\ 4)$.
2. $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$. Hence if $a \in \mathbb{Z}_p^*$, then $\left(\frac{a^2}{p}\right) = 1$.
3. If $a \equiv b$ $(mod\ p)$, then $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.
4. $\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}$. Hence $\left(\frac{2}{p}\right) = 1$ if $p \equiv 1$ or 7 $(mod\ 8)$, and $\left(\frac{2}{p}\right) = -1$ if $p \equiv 3$ or 5 $(mod\ 8)$.
5. (*law of quadratic reciprocity*) If q is an odd prime distinct from p, then

$$\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)(-1)^{(p-1)(q-1)/4}$$

.

In other words, $\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)$ unless both p and q are congruent to 3 modulo 4, in which $\left(\frac{p}{q}\right) = -\left(\frac{q}{p}\right)$.

The Jacobi symbol is a generalization of the Legendre symbol to integers n which are odd but not necessarily prime.

Definition 2.1.17 Let $n \geq 3$ be odd with prime factorization $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$. Then the *Jacobi symbol* $\left(\frac{a}{n}\right)$ is defined to be

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1}\left(\frac{a}{p_2}\right)^{e_2}\cdots\left(\frac{a}{p_k}\right)^{e_k}$$

.

Theorem 2.1.19 (*properties of Jacobi symbol*) Let $m \geq 3$, or $n \geq 3$, be odd integers, and $a, b \in \mathbb{Z}$. Then the Jacobi symbol has the following properties:

1. $\left(\frac{a}{n}\right) = 0, 1$ or -1. Moreover, $\left(\frac{a}{n}\right) = 0$ if and only if $gcd(a, n) \neq 1$.
2. $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$. Hence if $\mathbb{Z}_n^*$, then $\left(\frac{a^2}{n}\right) = 1$.
3. $\left(\frac{a}{mn}\right) = \left(\frac{a}{m}\right)\left(\frac{b}{n}\right)$.
4. If $a \equiv b \ (mod\ n)$, then $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$.
5. $\left(\frac{1}{n}\right) = 1$.
6. $\left(\frac{-1}{n}\right) = (-1)^{(n-1)/2}$. Hence $\left(\frac{-1}{n}\right) = 1$ if $n \equiv 1 \ (mod\ 4)$, and $\left(\frac{-1}{n}\right) = -1$ if $n \equiv 3 \ (mod\ 4)$.
7. $\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}$. Hence $\left(\frac{2}{n}\right) = 1$ if $n \equiv 1$ or $7 \ (mod\ 8)$, and $\left(\frac{2}{n}\right) = -1$ if $n \equiv 3$ or $5 \ (mod\ 8)$.
8. $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)(-1)^{(m-1)(n-1)/4}$. In other words $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)$ unless both m and n are congruent to 3 modulo 4, in which case $\left(\frac{m}{n}\right) = -\left(\frac{n}{m}\right)$.

2.2 Groups

This section provides an overview of basic algebra objects and their properties.

Definition 2.2.1 A *binary operation* $*$ on a set S is a mapping from $S \times S$ to S. That is, $*$ is a rule which assigns to each order pair of elements from S an element of S.

Definition 2.2.2 A *group operation* $(G, *)$ consists of a set G with a binary operation $*$ on G satisfying the following three axioms.

1. The group is a *associative*. That is, $a * (b * c) = (a * b) * c$ for all $a, b, c \in G$.
2. There is an element $1 \in G$, called the *identity element*, such that $a * 1 = 1 * a = a$ for all $a \in G$.
3. For each $a \in G$ there exists an element $a^{-1} \in G$, called the *inverse* of a, such that $a * a^{-1} = a^{-1} * a = 1$.

   A group G is *abelian* (or *commutative*) if, furthermore,

4. $a * b = b * a$ for all $a, b \in G$.

Definition 2.2.3 A group G is a *finite* if $|G|$ is finite. The number of elements in a finite group is called its *order*.

Definition 2.2.4 A group G is a *cyclic* if there is an element $g \in G$ such that for each $b \in G$ there is an integer i with $b = g^i$. Such an element g is called a *generator* of G.

Theorem 2.2.1 If G is a group an $a \in G$, then the set of all powers of a forms a cyclic subgroup of G, called the subgroup *generated by* a, and denoted $\langle a \rangle$.

Theorem 2.2.2 Let G be a group, and let $a \in G$ be an element of finite order t. Then $|\langle a \rangle|$, the size of the subgroup generated by a, is equal to t.

Example 2.2.1 Consider the multiplicative group $a \in \mathbb{Z}_{19}^* = \{1, 2, \ldots, 18\}$ of order 18. The group is cyclic, and a generator is $g = 2$. The subgroups of $a \in \mathbb{Z}_{19}^*$, and the generators are listed in the following table.

Subgroup	Generators	Order
$\{1\}$	1	1
$\{1, 18\}$	18	2
$\{1, 7, 11\}$	$7, 11$	3
$\{1, 7, 8, 11, 12, 18\}$	$8, 12$	6
$\{1, 4, 5, 6, 7, 9, 11, 16, 17\}$	$4, 5, 6, 9, 16, 17$	9
$\{1, 2, 3, \ldots, 18\}$	$2, 3, 10, 13, 14, 15$	18

Table 2.4: *The subgroups of* $\mathbb{Z}_{19}^*$.

2.3 Rings

Definition 2.3.1 A *ring* $(R, +, \times)$ consists of a set R with two binary operations arbitrarily denoted $+$ (addition) and $\times$ (multiplication) on R, satisfying the following axioms.

1. $(R, +)$ is an abelian group with identity denoted 0.
2. The operation $\times$ is associative. That is $a \times (b \times c) = (a \times b) \times c$ for all $a, b, c \in R$.
3. There is a multiplicative identity denoted 1, with $1 \neq 0$, such that $1 \times a = a \times 1 = a$ for all $a \in R$.
4. The operation $\times$ is *distributive* over $+$. That is, $a \times (b + c) = (a \times b) + (a \times c)$ and $(b + c) \times a = (b \times a) + (c \times a)$ for all $a, b, c \in R$.

The ring is a *commutative ring* if $a \times b = b \times a$ for $a, b \in R$.

Definition 2.3.2 An element of a of a ring R is called a *unit* or an *invertible element* if there is an element $b \in R$ such that $a \times b = 1$.

2.4 Fields

Definition 2.4.1 A *field* is a commutative ring in which all non-zero elements have multiplicative inverses.

Theorem 2.4.1 $\mathbb{Z}_n$ is a field (under the usual operations of addition and multiplication modulo n) if and only if n is a prime number. If n is prime, then $\mathbb{Z}_n$ has characteristic n.

Theorem 2.4.2 If the characteristic n of a field is not 0, then n is a prime number.

2.4.1 Finite Fields

Definition 2.4.2 A *finite field* is a field F which contains a finite number elements. The *order* of F is the number of elements in F.

Theorem 2.4.3 (*existence and uniqueness of finite fields*)

1. If F is a finite field, then F contains p^m elements for some prime p and integer $m \geq 1$.

2. For every prime power order p^m, there is a unique (up to isomorphism) finite field of order p^m . This field is denoted $\mathbb{F}_{p^m}$, or sometimes by $GF(p^m)$.

Theorem 2.4.4 if $\mathbb{F}_q$ is a finite field of order $q = p^m, p$ is a prime, then the characteristic of $\mathbb{F}_q$ is p. Moreover, $\mathbb{F}_q$ contains a copy of $\mathbb{Z}_p$ as a subfield. Hence $\mathbb{F}_q$ can be viewed as an extension field of $\mathbb{Z}_p$ of degree m.

Theorem 2.4.5 (*subfields of a finite field*) Let $\mathbb{F}_q$ be a finite field of order $q = p^m$. Then every subfield of $\mathbb{F}_q$ has order p^n, for some n that is a positive divisor of m. Conversely, if n is positive divisor of m, then there is exactly one subfield of $\mathbb{F}_q$ of order p^n; an element $a \in \mathbb{F}_q$ is in the subfield $\mathbb{F}_{p^n}$ if and only if $a^{p^n} = a$.

Definition 2.4.3 The non-zero elements of $\mathbb{F}_q$ from a group under multiplication called the *multiplicative group* of $\mathbb{F}_q$, denoted $\mathbb{F}_q^*$.

Theorem 2.4.6 $\mathbb{F}_q^*$ is a cyclic group of order $q - 1$. Hence $a^q = a$ for all $a \in \mathbb{F}_q$.

Proposition 2.4.1 The order of any $a \in \mathbb{F}_q^*$ devides $q - 1$.

Proof.
For $a^{q-1} = 1$ let d be the order of a, i.e., the smallest positive power which gives 1. If d did not divide $q - 1$, we could find a smaller positive number r- namely, the remainder when

$$q - 1 = bd + r, \quad where\ 1 \leq r < d$$

is divided by d- such that

$$a^r \cdot a^{bd} = a^{q-1} = 1.$$

But this contradicts the minimality of d. This concludes the proof.

Definition 2.4.4 A generator of the cyclic group $\mathbb{F}_q^*$ is called a *primitive element* or *generator* of $\mathbb{F}_q$.

Theorem 2.4.7 If $a, b \in \mathbb{F}_q$, a finite field of characteristic p, then

$$(a+b)^{p^t} = a^{p^t} + b^{p^t} \text{ for all } t \geq 0.$$

2.5 Finding Primitive Elements in $\mathbb{Z}_p$

In many public key distribution protocols based on Diffie-Hellman, it is necessary to find a primitive element of $g \in \mathbb{Z}_p$, where p is prime[16]. This is not too difficult to do if the factorization of $p-1$ is known. For the remainder of this section, let us assume that the factorization of $p-1$ is

$$p - 1 = {p_1}^{e_1} {p_2}^{e_2} \cdots {p_k}^{e_k}$$

where $p_1, p_2, \cdots, p_k$ are distinct primes.

First, we present a lemma that provides a method of determining if a given element $g \in \mathbb{Z}_p^*$ is primitive.

Lemma 2.5.1 Suppose p is prime, and the factorization of $p-1$ is as given above. Then $g \in \mathbb{Z}_p^*$ is a primitive.

$$g^{(p-1)/p_j} \not\equiv 1 \ (mod\ p)$$

for $1 \leq j \leq k$.

Proof.
Let d denote the order of g. We know that d is a divisor of $p-1$ and g is primitive if and only if $d = p-1$.

First, suppose that $g^{(p-1)/p_j} \equiv 1 \ (mod\ p)$ for some j. Then clearly $d \leq (p-1)/p_j$, so certainly $d \neq p-1$.

Conversely, suppose that $g^{(p-1)/p_j} \not\equiv 1 \ (mod\ p)$ for $1 \leq j \leq k$. Suppose that $d \neq p-1$. Since d is a divisor of $p-1$ and $d < p-1$, there exists a prime p_j $(1 \leq j \leq k)$ such that p_j is a divisor of $(p-1)/d$. But this implies that d is a divisor of $(p-1)/p_j$. Hence, it follows that

$$g^{(p-1)/p_j} \equiv g^d \equiv 1 \ (mod\ p),$$

which is a contradiction. This proves that $d = p-1$, as desired.

Now, given that we have an efficient method of detemining if a given element g is primitive, how do we go about finding a primitive element? This can be done quite easily by means of Las Vegas algorithm, by choosing random values for g and testing them, until a primitve element is found. The effectiveness of this approach depends on the probability that a random element $g \in \mathbb{Z}_p^*$ is primitive. Altogether, there are exactly $\varphi(p-1)$ primitive elements in $\mathbb{Z}_p^*$, so the probability that a random element g is a primitive element is $\varphi(p-1)/(p-1)$.

One special case of interest is when $p = 2q + 1$, where q is prime. In this case, the following corollary is obtained.

Corollary 2.5.1 Suppose p and q are prime, and $p = 2q + 1$. Suppose $g \in \mathbb{Z}_p^*$ and $g \not\equiv \pm 1 \ (mod\ p)$. Then g is a primitive element if and only if $g^{(p-1)/2} \not\equiv 1 \ (mod\ p)$.

Proof.
Observe that $g^{(p-1)/q} \equiv g^2 \ (mod\ p)$, and $g^2 \equiv 1 \ (mod\ p)$ if and only if $g \equiv \pm 1 \ (mod\ p)$. Hence the result follows from the last Lemma.

In fact, If $g \not\equiv \pm 1 \ (mod\ p)$ and g is not primitive, then $g^{(p-1)/2} \equiv 1 \ (mod\ p)$. But then we have Thus, by this Corollary $(-g)$ must be primitive.

$$\begin{aligned} (-g)^{(p-1)/2} &\equiv (-1)^{(p-1)/2} g^{(p-1)/2} \ (mod\ p) \\ &\equiv (-1)^{(p-1)/2} \ (mod\ p) \\ &\equiv -1 \ (mod\ p). \end{aligned}$$

This result is recorded as follows:

Corollary 2.5.2 Suppose p and q are prime, and $p = 2q + 1$. Suppose $g \in \mathbb{Z}_p^*$ is not a primitive element, and $g \not\equiv \pm 1 \ (mod\ p)$. Then $(-g)$ is a primitive element.

This means that we have an efficient deterministic algorithm to find a primitive element for when p and $(p-1)/2$ are both prime.

It is not so easy to verify that elements are primitive if the factorization of $p - 1$ is not known. For this reason, the designer of a cryptosystem will often construct p in such a way that the factorization of $p - 1$ is known. For example, it is often desirable to implement a cryptosystem in $\mathbb{Z}_p$, where $p = 2q + 1$ and p and q are both prime. One reason why this might be done is that it ensures that the system will not be vulnerable to a Pohlig-Hellman[20] attack on the discrete logarithm problem. To find such a p, the designer of the system will choose a random odd value q, and test both p and $p = 2q + 1$ for primality using one of the probablistic primality tests[1,13]. If either of p or q is found to be composite, then a new random value of q is chosen and the process is repeated.

As another example, several protocols are implemented in $\mathbb{Z}_p$ where $p - 1$ has a prime divisor q of a specified size. A convenient realization of such a system would be to take $p = 2qr + 1$, where p, q and r are all primes. if q is to be a 160-bit prime and p is to be a 512-bit prime, then r will be a prime of approximately 352 bits. Here, the designer of the system would begin by choosing random values q and r of the appropriate size, and then

define $p = 2qr + 1$. The three integers p, q and r will all be tested for primality using a probabilistic primality testing algorithm.

2.6 The Pohlig-Hellman Algorithm

In this section, a bit of further explanation concerning the workings of the Pohlig-Hellman algorithm is provided. It is needed to be understood this algorithm because Pohlig-Hellman attack is a serious attack which is based on mathematical views. The same notation will be used as in the last section: p is prime, g is a primitive element in $\mathbb{Z}_p$, and $h \in \mathbb{Z}_p^*$. Our goal is to determine $a = \log_g h$, where, without loss of generality, $0 \leq a \leq p - 2$.

The prime power factorization of $p - 1$ is

$$p - 1 = {p_1}^{e_1} {p_2}^{e_2} \cdots {p_k}^{e_k},$$

where $p_1, p_2, \cdots, p_k$ are distinct primes. The main step is to compute $g \ (mod \ {p_i}^{e_i})$, $1 \leq i \leq k$. So Suppose that $q = p_i$ and $e = e_i$ for some i, $1 \leq i \leq k$. Here, it will be showed how to compute $x = a \ (mod \ q^e)$.

First, x is expressed as

$$x = \sum_{i=0}^{e-1} a_i q^i,$$

where $0 \leq a_i \leq q - 1$ $(0 \leq i \leq e - 1)$. From this it follows that

$$a = a_0 + a_1 q = \ldots + a_{e-1} q^{e-1} + sq^e,$$

for some integer s.

The computation of a_0 follows from the fact that

$$h^{\frac{p-1}{q}} \equiv g^{\frac{a_0(p-1)}{q}} \ (mod \ p). \tag{2.1}$$

Here is a proof of Equatian (2.1):

$$\begin{aligned}
h^{\frac{p-1}{q}} &\equiv (g^a)^{\frac{(p-1)}{q}} \ (mod \ p) \\
&\equiv (g^{a_0 + a_1 q + \ldots + a_{e-1} q^{e-1} + sq^e})^{\frac{(p-1)}{q}} \ (mod \ p) \\
&\equiv (g^{a_0 + Kq})^{\frac{(p-1)}{q}} \ (mod \ p) \qquad \text{(Where } K \text{ is an integer)} \\
&\equiv g^{\frac{a_0(p-1)}{q}} g^{K(p-1)} \ (mod \ p) \\
&\equiv g^{\frac{a_0(p-1)}{q}} \ (mod \ p).
\end{aligned}$$

From this,it is a simple matter to determine a_0.

The next step would be to compute $a_1, a_2, \ldots, a_{e-1}$ (if $e > 1$). These computations can be done from a suitable generalization of Equation (2.1).

First denote $h_0 = h$, and

$$h_j = hg^{-(a_0+a_1q+\cdots+a_{j-1}q^{j-1})} \ (mod\ p),$$

for $0 \leq j \leq e-1$. We make use of the following generalization of Equation (2.1):

$$(h_j)^{\frac{p-1}{q^{j+1}}} \equiv g^{\frac{a_j(p-1)}{q}} \ (mod\ p). \tag{2.2}$$

(Observe that when $j = 0$, Equation (2.2) reduces to Equation (2.1).)

The proof of Equation (2.2) is much the same as that Equation (2.1):

$$\begin{aligned}
(h_j)^{\frac{p-1}{q^{j+1}}} &\equiv (g^{a-(a_0+a_1q+\ldots+a_{j-1}q^{j-1})})^{\frac{p-1}{q^{j+1}}} \ (mod\ p) \\
&\equiv (g^{a_jq^j+\cdots+a_{e-1}q^{e-1}+sq^e})^{\frac{p-1}{q^{j+1}}} \ (mod\ p) \\
&\equiv (g^{a_jq^j+K_jq^{j+1}})^{\frac{p-1}{q^{j+1}}} \ (mod\ p) \qquad \text{(Where } K_j \text{ is an integer)} \\
&\equiv g^{\frac{a_j(p-1)}{q}} g^{K_j(p-1)} \ (mod\ p) \\
&\equiv g^{\frac{a_j(p-1)}{q}} \ (mod\ p).
\end{aligned}$$

Hence, given h_j, it is straightforward to compute a_j from Equation (2.2).

To complete the description of the algorithm, it suffices to observe that h_{j+1} can be computed from h_j by means of a simple recurence relation, once a_j is known. This follows from the following relation, which is proved easily:

$$h_{j+1} = h_j g^{-a_jq^j} \ (mod\ p). \tag{2.3}$$

Now, we can compute $a_0, h_1, a_1, h_2, \cdots, h_{e-1}, a_{e-1}$ by applying Equation (2.2) and (2.3).

Chapter 3

Public Key Distribution System

In a communication network, two users, wishing to communicate with each other via a common-key encryption scheme, have to share a secret cryptographic key not known to other users. This is what we call the key distribution problem[19].

This chapter considers key establishment protocols and related cryptographic techniques which provide shared secrets between to parties, typically for subsequent use as symmetric keys for a variety of cryptographic purposes including encryption, decryption, and message authentication. Remainder of this chapter is organized as follows. Section 1 provides background materials including basic definitions and concepts, and discussion of objective. Section 2 and section 3 discuss one-way functions and key agreement protocols, respectively, based on Diffie-Hellman Key Exchange Protocols.

3.1 Definitions

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

There are several ways in which cryptosystems can be classified. Generally, cryptosystems are classified into two classes; *Private-Key* cryptosystems which are beyond this research, and *Public-Key* cryptosystems which will be discussed in whole this research.

A cryptosystem is a five-tuple ($\mathcal{M}$, $\mathcal{C}$, $\mathcal{K}$, $\mathcal{E}$, $\mathcal{D}$), where the following conditions are satisfied:

- $\mathcal{M}$ denotes a set called the *message space*. An element of $\mathcal{M}$ is called a *plaintext message* or simply a *plaintext*.
- $\mathcal{C}$ denotes a set called the *ciphertext space*. An element of $\mathcal{C}$ is called a *ciphertext*.
- $\mathcal{K}$ denotes a set called the *key space*. An element of $\mathcal{K}$ is called a *key*.
- $\mathcal{E}$ denotes a set called the *enciphering transformation*,

$$\mathcal{E} = \{E_k \mid k \in \mathcal{K}\}$$
$$E_k \ : \ \mathcal{M} \to \mathcal{C}, \qquad (k \in \mathcal{K}).$$

- $\mathcal{D}$ denotes a set called the *deciphering transformation*,

$$\mathcal{D} = \{D_k \mid k \in \mathcal{K}\}$$
$$D_k \ : \ \mathcal{C} \to \mathcal{M}, \qquad (k \in \mathcal{K}).$$

An enciphering scheme consists of a set $\{E_e \mid e \in \mathcal{K}\}$ of encryption transformations such that $E_e(m) = c$ for all $m \in \mathcal{M}$, and a corresponding set $\{D_d \mid d \in \mathcal{K}\}$ of deciphering transformations such that $D_d(c) = m$ with the property that for each $e \in \mathcal{K}$ there is a unique $d \in \mathcal{K}$ such that $D_e = E_e^{-1}$ that is, $D_d(E_e(m)) = m$ for all $m \in \mathcal{M}$.
The keys e and d in the proceding definition are referred to as a *key pair* and sometimes denoted by (e, d). Note that, e and d could be the same.

3.2 One-Way Functions

Definition 3.2.1 A function f from a set X to a set Y is called a *one-way function* if $f(x)$ is " easy" to compute for all $x \in X$ but " essentially all" elements $y \in Im(f)$ it is " computationally infeasible" to find any $x \in X$ such that $f(x) = y$.

The notion of one-way functions is at the core of public-key cryptography which has the property that someone who knows only how to encipher cannot use the enciphering key to find the deciphering function, becuase $f \ : \ \mathcal{M} \stackrel{easy}{\longrightarrow} \mathcal{C}$ is easy to compute once the enciphering key E_k is known, but it is very hard in practice to compute the inverse function $f^{-1} \ : \ \mathcal{C} \stackrel{hard}{} \mathcal{M}$. That is, from the standpoint of realistic computability, the function f is not invertible (without some additional information - the deciphering key D_k). Such a function f is called a *trapdoor function*. That is, a trapdoor function f is a function without having some additional auxiliary information beyond what is necessary to compute f. The inverse f^{-1} is easy to compute, however, for someone who has this information D_k (the " decipheting key"). This notion should not be confused with functions that are mathimatically non-invertible for lack of being ono-to-one or onto. More details about functions can be found in references [7,12].

Example 3.2.1 A simple example of a candidate one-way function is *integer multiplication*. It is easy to multiply very large integers whereas even the most powerful computer with the best available algorithm is incapable of factoring a mere two hundred digit number that is the product of two roughtly equal size within a reasonable time.

Example 3.2.2 A *prime number* is a positive integer greater than 1 whose only positive integer divisors are 1 and itself. Select primes $p = 48611$, $q = 53993$, form $n = pq = 2624653723$, and let $X = \{1, 2, 3, \ldots, n-1\}$. Define a function f on X by $f(x) = r_x$ for each $x \in X$, where r_x is the remainder when x^3 is devided by n. For instance, $f(2489991) = 1981394214$ since $2489991^3 = 5881949859 \cdot n + 1981394214$. Computing

$f(x)$ is relatively simple thing to do, but to inverse the procedure is much more difficult; that is, given a remainder to find the value x which was orginally cubed. This procedure is referred to as the computation of a modular cube root with modulus n. If the factors of n are unknown and large, this is a difficult problem; however, if the factors p and q of n are known then there is an efficient algorithm for computing modular cube roots.

3.3 Authentication and Identification

Authentication protocols are designed to provide two or more specific entities communicating over an open network to achieve some cryptographic goals such as confidentiality, data integrity, entity authentication, message authentication, not repudiation, and key authentiation. In this section a brief introduction to authentication is discussed, for details refere to [6].

An authenticaiton is one of the most inportant of all informaion security objectives. As discussed in [14] it must be easy for anyone to recognize the signature as authentic, but impossible for anyone other than the legitimate signer to produce it.

In public key cryptography there is an especially easy way to identify oneself in such a way that no one could be simply pretending to be you. Let A (Alice) and B (Bob) be two users of the system. Let f_A be the enciphering transformation with which any user of the system sends a message to Alice, and let f_B be the same for Bob. For simplicity, we assume that the set $\mathcal{M}$ of all possible plaintext message units and the set $\mathcal{C}$ of all possible ciphertext message units are equal, and are the same for all users. Let M be Alice's " signature" (perhaps including an identification number, a statement of the time the message was sent, ets.). It would not be enough for Alice to send Bob the encoded message $f_B(M)$, since *everyone* knows how to do that, so there would be no way of knowing that the signature was not forged. Rather, at the beginning (or end) of the message Alice transmits $f_B {f_A}^{-1}(M)$. Then, when Bob deciphers the whole message, including this part, by applying ${f_B}^{-1}$, he finds that everything has become plaintext except for a small section of jibberish, which is ${f_A}^{-1}(M)$. Since Bob knows that the message is claimed to be from Alice, he applies f_A (which he knows, since Alice's encihpering key is public), and obtains M. Since no one other than Alice could have applied the function ${f_A}^{-1}$ which is inverted by f_A, he knows that the message was from Alice.

3.3.1 Identification

Definition 3.3.1 An *identification* or *entity authentication* technique assures one party of both the identity of a second party involved, and that the second was active at the time the evidence was created or acquired.

Typically the only data transmitted is that necessary to identify the communication parties. The entities are both active in the communication, giving a timliness guarantee.

3.3.2 Data origin authentication

Definition 3.3.2 *Data origin authentication* or *message authentication* techniques provide to one party which receives a message assurance of the identity of the party which originated the message.

Often a message is provided to receiver along with additional information so that the receiver can determine the identity of the entity who originated the message. This form of authentication typically provides no guarantee of timeliness, but is useful in situations where one of the parties is not active in the communication.

3.4 Diffie-Hellman and related Key Agreement Protocols

Key establishment protocols come in various flavors. In *key transport* protocols, a key is created by one entity and securely transmitted to the second entity, while in *key agreement* protocols both entities contribute information which is used to derive the shared secret key. In *symmetric* protocols the two entities a priori possess common secret information, while in *asymmetric* protocols the two entities share only public informaiton that has been authenticated. This research is concerned with two-party authenticated key agreement protocols in the asymmetric setting.

The design of asymmetric authenticated key agreement protocols has a checkered history. Over the years, numerous protocols have been proposed to meet a variety of desirable security and performance requirements. Many of these protocols were subsequently found to be flawed, and then either were modified to resist the new attacks, or were totally abandoned. After a series of attacks and modifications, only those surviving protocols which had received substantial public security and were believed to resist all known attacks were deemed secure for practical usage [18].

This section focuses on asymmetric authentication key agreement protocols whose security is based on intractability of the Diffie-Hellman problem. Next section says about the idea why discrete logs are generally used.

3.4.1 Why discrete logs and Diffie-Hellman?

Almost everything that public key cryptography provides, such as digital signatures and key exchange, can be accomplished with RSA and its variants. However, cryptosystems based on discrete exponentiation remain of interest for three main reasons [26]:

1. **Patent issues** The Diffie-Hellman patent expired in 1997. Therefore anyone interested in using public key cryptography in the United States (which is the only place where this patent was applied for and issued) can save money and also avoid licensing negotiations.

2. **Technical advantages** In many cases where algorithms of comparable functionality exist, say one over the finite field of integers modulo a prime p, and another

using a composite integer n of the same size, breaking the discrete log modulo p appears to be somewhat harder than factoring the integer n.
Some other advantages of discrete log cryptosystems come from their limitations. It is widely believed that the U.S. Digital Signature Algorithm is based on discrete logs because it is harder to use it for encryption than if it were based on RSA (and thus on integer factorization). This helps enforce export control regulations on strong encryption without weakening the digital signature methods that are less stringently controlled. On the other hand, many people like the Diffie-Hellman algorithm, since the session key it generates is evanescent. In the simplest application of RSA to key generation, Alice creates a session key and transmits it to Bob using Bob's public key. An eavesdropper who can coerce Bob afterwards into revealing his private key can then recover the full text of the communication exchanged by Alice and Bob. In contrast, if Alice and Bob use Diffie-Hellman to generate the session key, destroy it after the session ends, and do not store their communication, then neither coercion nor cryptanalysis will enable the eavesdropper to find out what information was exchanged.

3. They are different. Cryptographers have learned by bitter experience that it is unwise to put all eggs in a single basket. It is desirable to have a diversity of cryptosystems, in case one is broken. It is an unfortunate fact that discrete logs and integer factorization are so close that many algorithms developed for one problem can be modified to apply to the other. For security, it would be better to have much more diversity. However, more than two decades after the publication of the first two practical public key systems, the Diffee-Hellman and the RSA algorithms, the only public key cryptosystems that are trusted and widely deployed are based on the presumed difficulty of the same two problems those schemes relied upon. There have been many attempts to find public key schemes based on other principles, but so far most have led to systems that were broken, and the ones that are still standing are often regarded with suspicion.

The following notation is used throughout this chapter and the next.

Alice, Bob	Honest entities.
Lucy	Attacker (also called man-in-the-middle attack).
p	A prime number.
g	A primitive element of $\mathbb{Z}_p^*$.
x, y	Static private keys of Alice and Bob.
X, Y	Static public keys of Alice and Bob; $X \equiv g^x \ (mod\ p), Y \equiv g^y \ (mod\ p)$.
r, r'	Ephemeral private keys of Alice and Bob.

The *domain* parameters (p, g) are common to all entities.

3.4.2 Diffie-Hellman Key Agreement

Diffie-Hellman key agreement provided the first practical solution to the key distribution problem, allowing two parties, never having met in advance or shared keying material, to establish a shared secret by exchanging messages over an open channel. The security on the intractability of the Diffie-Hellman problem (DHP) which will be discussed a bit later.

Protocol 1 Ephemeral Diffie-Hellman Key Exchange

Both users Alice and Bob first agree on a prime p and a primitive root $g \in \mathbb{Z}_p^*$.

1. Alice chooses r at random, $1 \leq r \leq p-2$.
2. Alice computes $X \equiv g^r \ (mod\ p)$ and sends it to Bob.
3. Bob chooses r' at random, $1 \leq r' \leq p-2$.
4. Bob computes $Y \equiv g^{r'} \ (mod\ p)$ and sends it to Alice.
5. Alice computes
$$K \equiv Y^r = (g^{r'})^r = g^{rr'} \ (mod\ p).$$
6. Bob computes
$$K' \equiv X^{r'} = (g^r)^{r'} = g^{rr'} \ (mod\ p).$$

While the epheremal Diffie-Hellman protocol provides implicit key authentication in the presence of passive adversaries, it does not on its own provide any useful services in the presence of active adversaries since neither entity is provided with any assurance regarding the identity of the entity it is communicating with. Ephemeral Diffie-Hellman Key Exchange is supposed to look like this:

$$Alice \quad \xrightarrow{\quad g^r \quad} \quad \xleftarrow{\quad g^{r'} \quad} \quad Bob$$

Protocol 2 Static Diffie-Hellman Key Exchange

Here, it is assumed that static public keys are exchanged via certificates. $Cert_{Alice}$ denotes Alice's public key certificate, containing a string of information that uniquely identifies Alice's static public key X.

1. Alice sends $Cert_{Alice}$ to Bob.

2. Bob sends $Cert_{Bob}$ to Alice.

3. Alice computes $K \equiv Y^x = (g^y)^x = g^{xy} \ (mod\ p)$.

4. Bob computes $K' \equiv X^y = (g^x)^y = g^{xy} \ (mod\ p)$.

Since each entity is assured that it possesses an authentic copy of the other entity's public key, the static Diffie-Hellman protocol provides implicit key authentication. A major drawback, however, is that Alice and Bob compute the same shared secret $K = K' \equiv g^{xy}$ for each run of protocol. Static Diffie-Hellman Key Exchange is supposed to look like this:

$$Alice \quad \xrightarrow{g^x \in Cert_{Alice}} \quad \xleftarrow{g^y \in Cert_{Bob}} \quad Bob$$

The drawbacks of the ephemeral and static Diffie-Hellman protocols can be alleviated by using both static and ephemeral keying materail in the formation of shared secrets which will be discussed a bit later in this chapter.

Note 3.4.1 (*control over Diffie-Hellman key*) While it may appear as though Diffie-Hellman key agreement allows each party to guarantee key freshness and preclude key control, use of an exponential with small multiplicative order restricts the order of the overall key. The most degenerate for $\mathbb{Z}_p$ would be selection of 0 as private exponent, yielding an exponential with order 1 and the multiplicative identity itself as resulting key. Thus, either participant may force the resulting key into a subset of the original range set. Relatedly, some variants of Diffie-Hellman involving unauthenticated exponentials are vulnerable to the following active attack. Assume $g \in \mathbb{Z}_p^*$ where $p = Rq + 1$ (consider $R = 2$ and q prime). Then $\beta = g^q = g^{(p-1)/R}$ has order R ($\beta = -1$ for $R = 2$). If Alice and Bob exchange unauthenticated short-term exponential g^r and $g^{r'}$, an active adversary may replace these by $(g^r)^q$ and $(g^{r'})^q$, forcing the share key to be $K = g^{rr'q} = \beta^{rr'}$, which takes one of only R values ($+1$ or -1 for $R = 2$). K may thus be found by exhaustive trial of R values. A more direct attack involves simply replacing the exchange exponential by $+1$ or $p - 1 = -1$. This general class of attacks may be prevented by authenticating the exchanged exponentials[5].

Both parties Alice and Bob can encrypt messages using the following encryption transformaion,

$$c \equiv m^K \ (mod\ p).$$

In order to decrypt, the receiver first finds the deciphering key $\overline{K}$ via the congruence,

$$K \cdot \overline{K} \equiv 1 \ mod\ (p - 1).$$

and then calculates the message,

$$m \equiv c^{\overline{K}} \ (mod\ p).$$

Note that, $\overline{K}$ exists if and only if $gcd(K, p-1) = 1$.

We illustrate the Diffie-Hellman system in the example given below.

Example 3.4.1 Assume the modulus $p = 47$ and the primitive element $g = 23$. Suppose that Alice and Bob have selected their secret keys $x = 12$ and $y = 33$. In order to fix the common secret key K, they calculate their partial keys :

$$X \equiv g^x = 23^{12} = 27 \ (mod\ 47).$$
$$Y \equiv g^y = 23^{33} = 33 \ (mod\ 47).$$

After they exchange their partial keys, Alice and Bob compute the common secret key,

$$K \equiv Y^x = X^y = 27^{33} = 25 \ (mod\ 47).$$

They also find the secret deciphering key $\overline{K}$ using the following congurence :

$$K \cdot \overline{K} \equiv 1 \quad (mod\ p-1) \rightarrow \overline{K} \equiv 35 \ (mod\ 46).$$

Now, if the message is $m = 16$, then the cryptogram is :

$$c \equiv m^K = 16^{25} = 21 \ (mod\ 47).$$

The receiver recreates the message as following :

$$m \equiv c^{\overline{K}} = 21^{35} = 16 \ (mod\ 47).$$

Unfortunately, the protocol is vulnerable to an active adversary who uses a *man-in-the-midle attack*. There is an episode of *The Lucy Show* in which Vivian Vance is having dinner in a restaurant with a date, and Lucille Ball is hiding under the table. Vivian and her date decide to hold hands under the table. Lucy, trying to avoid detection, holds hands with each of them and they think they are holding hands with each other.

A man-in-the-middle attack on the Diffie-Hellman protocol works in the same way. Lucy will intercept messages between Alice and Bob and substitute her own messages, as indicated in the following diagram :

$$Alice \ \underset{g^{y'}}{\overset{g^x}{\longrightarrow \atop \longleftarrow}} \ Lucy \ \underset{g^{y}}{\overset{g^{x'}}{\longrightarrow \atop \longleftarrow}} \ Bob$$

At the end of the protocol, Alice has actually established the secret key $g^{xy'}$ with Lucy, and Bob has established a secret key $g^{x'y}$ with Lucy. When Alice tries to encrypt a message to send to Bob, Lucy will be able to decrypt it but Bob will not. (A similar situation holds if Bob sends a message to Alice.)

Clearly, it is essensial for Alice and Bob to make sure that they are exchanging messages with each other and not with Lucy. Before excahnging keys, Alice and Bob might carry out a separate protocol to establish each others's identity. But this offers no protection against an active adversary in the man-in-the-middle attack if Lucy simply remains inactive until after Alice and Bob have proved their identities to each other. We will discuss more on this case after introducing some ther protocols which are based on Diffie-Hellman problem.

Diffie-Hellman Problem

The *Diffie-Hellman Problem* is closely related to the *Discrete Logarithm Problem*. It is of significance to public-key cryptosystem becuase its apparent intractability forms the basis for the security of many cryptographic schemes including Diffie-Hellman Key Exchange.

Definition 3.4.1 The Diffie-Hellman Problem is the following : given a prime p, a generator of $\mathbb{Z}_p^*$, and elements $g^x \ (mod\ p)$ and $g^y \ (mod\ p)$, find $g^{xy} \ (mod\ p)$.

If an active adversary in the man-in-the-middle attack like Lucy could determin x from X, or if he could determine y from Y, then he could compute K exactly as Alice (or Bob) does. Both these computations are instances of Discrete Logaritm Problem. So, provided that the Discrete Logaritm Problem in $\mathbb{Z}_p^*$ is intractable, Diffie-Hellman Key Exchange is secure against this particular type of attack. However, it is an unproven conjecture that any algorithm that solves the Diffie-Hellman protocol could also be used to solve the Discrete Logaritm Problem.

By the remarks made above, the Diffie-Hellman Problem is no more difficult that the Discrete Logaritm Problem. Although we cannot say precisely how difficult this problem is.

3.4.3 ElGamal Key Agreement in one-pass

ElGamal key agreement is a Diffie-Hellman variant providing a one-pass protocol with unilateral key authentication, provided the public key of the recipient is known to the originator *a priori*. The protocol is more simply Diffie-Hellman key agreement wherein the public exponential of the recipient is fixed and has verifiable authenticity.

Protocol ElGamal key agreement (half-certified Diffie-Hellman)
Both users Alice and Bob first agree on a prime p and a primitive root $g \in \mathbb{Z}_p^*$.

1. Alice obtains an authentic copy of Bob's public key (p, g, g^y).
2. Alice chooses r at random, $1 \leq r \leq p-2$.
3. Alice computes $g^r \ (mod\ p)$ and sends it to Bob.

4. Alice computes $K \equiv (g^y)^r \ (mod\ p)$.

5. Bob computes $K' \equiv (g^r)^y \ (mod\ p)$.

Remark 3.4.1 (*assurances in one-pass ElGamal*) The recipient in the above protocol has no corroboration of whom he or she shares the secret key with, nor any key freshness assurances. Neither party obtains entity authentication or key confirmation. It means that even if an adversary in a man-in-the-middle-attack changes the values sent by Alice, Bob can not confirm that the received values are sent by Alice.

In the next section some protocols will be discussed which can alleviate the drawbacks of the ephemeral and static Diffie-Hellman protocols by using both static and ephemeral keying material in the formation of shared secrets.

3.4.4 MTI two-pass Key Agreement Protocols

In a cryptosystem it is essential to change the keys from time to time. In Diffie-Hellman Key Exchange scheme it is not so easy to change the public key in order to change the common key, because when a public key is registered in a public file it is hard to change it because of time consuming to prove that you are the right person. In order to this, there are some schemes that make it possible to change the common keys without changing the public keys. These schemes are classified to **MTI**[19] constructed by Matsumoto, Takashima and Imai which are interesting key agreement protocols by modifying Diffie-Hellman key Exchange. We present some of these protocols and consider the man-in-middle attack about one of them. Note that, MTI protocols were designed to provide implicit key authentication, and do not provide key confirmation and MTI/C0 protocol does not provide implicit key authentication at all.

Before describing any MTI schemes, let consider the following generalization. Any MTI protocol consists of 3 phases as following :

(a) Registration Phase

Each user $\langle i \rangle$ selects a secret data X_i and computes $Y_i \equiv g^{X_i} \ (mod\ p)$ and registers Y_i to the public file.

(b) Transfer Phase

If a user $\langle i \rangle$ wants to share a common data with another user $\langle j \rangle$, $\langle i \rangle$ transfers to $\langle j \rangle$ a data Z_{ij} gererated from a secret random number R_i and the registered data Y_j and/or the secret data X_i and/or the primitive element g of $GF(p)$. Then the user $\langle j \rangle$ sends back to $\langle i \rangle$ a similar data Z_{ji}. These Z_{ij} and Z_{ji} are called the " transferred data ".

(c) Key-Generation Phase

The user $\langle i \rangle$ composes a data K_{ij} from the accepted Z_{ji}, previously generated X_i and R_i, and the registered data Y_j. The user $\langle j \rangle$ composes a data K_{ji} in the same manner. The

data K_{ij} and K_{ji} are the same and denoted by K and said to be the "shared data ". This K will be used as the work key.

Here, both Alice and Bob keep secret the secret key x and y, and register $X \equiv g^x \ (mod\ p)$, $Y \equiv g^y \ (mod\ p)$ to a public file. The prime number p and its primitive root g of $\mathbb{Z}_p^*$ are public.

Protocol 1 MTI/A0 Key Agreement

1. Alice chooses a random secret r, $1 \leq r \leq p-2$.
2. Alice computes $Z \equiv g^r \ (mod\ p)$ and sends it to Bob.
3. Bob chooses a random secret r', $1 \leq r' \leq p-2$.
4. Bob computes $Z' \equiv g^{r'} \ (mod\ p)$ and sends it to Alice.
5. Alice computes
$$K \equiv Z'^x \cdot Y^r = g^{xr'+yr} \ (mod\ p).$$
6. Bob computes
$$K' \equiv Z^y \cdot X^{r'} = g^{xr'+yr} \ (mod\ p).$$

The information transmitted during the protocol is depicted as follows :

$$Alice \quad \xrightarrow{g^r \ mod\ p} \quad \xleftarrow{g^{r'} \ mod\ p} \quad Bob$$

Protocol 2 MTI/B0 Key Agreement

1. Alice chooses a random secret r, $1 \leq r \leq p-2$.
2. Alice computes $Z \equiv Y^r = g^{yr} \ (mod\ p)$ and sends it to Bob.
3. Bob chooses a random secret r', $1 \leq r' \leq p-2$.
4. Bob computes $Z' \equiv X^{r'} = g^{xr'} \ (mod\ p)$ and sends it to Alice.
5. Alice computes
$$K \equiv Z'^{\overline{x}} \cdot g^r = g^{r+r'} \ (mod\ p).$$
6. Bob computes
$$K' \equiv Z^{\overline{y}} \cdot g^{r'} = g^{r+r'} \ (mod\ p).$$

Protocol 3 MTI/C0 Key Agreement

1. Alice chooses a random secret r, $1 \leq r \leq p-2$.
2. Alice computes $Z \equiv Y^r = g^{yr} \ (mod\ p)$ and sends it to Bob.
3. Bob chooses a random secret r', $1 \leq r' \leq p-2$.
4. Bob computes $Z' \equiv X^{r'} = g^{xr'} \ (mod\ p)$ and sends it to Alice.
5. Alice computes
$$K \equiv Z'^{\overline{x}r} = g^{rr'} \ (mod\ p).$$
6. Bob computes
$$K' \equiv Z^{\overline{y}r'} = g^{rr'} \ (mod\ p).$$

Protocol 4 MTI/C1 Key Agreement

1. Alice chooses a random secret r, $1 \leq r \leq p-2$.
2. Alice computes $Z \equiv Y^{rx} = g^{rxy} \ (mod\ p)$ and sends it to Bob.
3. Bob chooses a random secret r', $1 \leq r' \leq p-2$.
4. Bob computes $Z' \equiv X^{r'y} = g^{r'xy} \ (mod\ p)$ and sends it to Alice.
5. Alice computes
$$K \equiv Z'^{r} = g^{xyrr'} \ (mod\ p).$$
6. Bob computes
$$K' \equiv Z^{r'} = g^{xyrr'} \ (mod\ p).$$

As we discussed before there is no protection against an active adversary in a man-in-the-middle attack. Clearly even with these protocols neither Alice nor Bob cannot confirm whom they have exchanged the keys. The only advantage of these protocols is that an active adversary like Lucy cannot intercept messages with Alice or Bob. But as well as Lucy, none of Alice or Bob can generate a right key to encipher or decipher the messages.

Let's look at the security of MTI/A0 protocol. In the man-in-the-middle attack it is possible for an active adversary like Lucy to alter the values that Alice and Bob send to each other. We depict one typical senario that might arise, as follows :

$$Alice \ \underset{g^{R'}}{\overset{g^{r}}{\longrightarrow\!\!\longleftarrow}} \ Lucy \ \underset{g^{r'}}{\overset{g^{R}}{\longrightarrow\!\!\longleftarrow}} \ Bob$$

In this situation, Alice and Bob will compute different keys:
Alice will compute

$$K \equiv g^{ry+R'x} \quad (mod\ p).$$

while Bob will compute

$$K' \equiv g^{Ry+r'x} \quad (mod\ p).$$

Remark 3.4.2 Neither of the shared keys computed by Alice or Bob can be carried out by Lucy, since they require knowledge of the secret exponents x and y, respectively. So even though Alice and Bob have computed different keys (which will of course be useless to them), neither of these keys can be computed by Lucy. In other words, both Alice and Bob are assured that the other is the only user in the network that could compute the key that they have computed. Let me say that it is good but not strong for some cases. If we can extend these schemes mentioned above, Alice or Bob can confirm whether the generated key is true or not.

Remark 3.4.3 (*computational complexity of MTI protocols*) The A0 and B0 protocols require 3 exponentiations by each party, whereas the C0 and C1 protocols require only 2. C1 has the additional advantage over B0 and C0 that no inverses are needed; however, these fixed long-term values may be precomputed.

Chapter 4

A Proposed Public Key Distribution Protocol

Numerous Diffie-Hellman-based protocols have been proposed over the years; however many have subsequently been found to have security flaws. A secure protocol should be able to withstand both *passive attacks* (where an adversary attempts to prevent a protocol from achieving its goals by merely observing honest entities carrying out the protocol) and *active attacks* (where an adversary additionally subverts the communications by injecting, deleting, altering or replaying messages), giving implicit key authentication and key confirmation as well.

This is a chapter that we will focus on the design of a new two-pass protocol. The security of the protocol is based on intractability of Diffie-Hellman Problem. As you notice it is an extension of the methods discussed in chapter 3 as MTI. In the last section of this chapter we will consider the security problems from various aspects. Before having begun the section let review some mathematical facts about the commutativity of power functions which are based on the computational difficulty of discrete logarithm.

Let h denote a non-zero element of a finite field $\mathbb{Z}_p$, where p is a prime number. Note that the power functions over $GF(p)$ have the following properties :

1. $(h^x)^y = (h^y)^x = h^{xy}$ (commutativity),
2. $h^x * h^y = h^{x+y}$ (homomorphic property),
3. $h^x = h^{x \ (mod \ p-1)}$,

where x and y denote arbitrary integers. From (3), we regard the exponents to be in the set $\{1, 2, \ldots, p-1\}$. for each x, $\overline{x}$ denote the multiplicative inverse of x modulo $(p-1)$, and it exists if $gcd(x, p-1) = 1$.

4.1 Protocol Description

This protocol consists of 3 phases; Registration Phase, Transfer Phase, and Key-Generation with Key-Verification Phase for recipient of the message. Each user like Alice and Bob se-

lects a secret data x and y relatively such that $2 \leq x, y \leq p-2$, computes $X \equiv g^x \ (mod\ p)$, and $Y \equiv g^y \ (mod\ p)$ and registers X, and Y to the public file. The prime number p and its primitive root g are public. Clearly, X and Y are public too.
For transferring data and key generating each user, Alice and Bob should do the following:

1. Alice chooses a random secret r, $2 \leq r \leq p-2$.
2. Alice computes $K \equiv Y^{rx} = g^{rxy} \ (mod\ p)$ as the shared key.
3. Bob chooses a random secret r', $2 \leq r' \leq p-2$, such that $gcd(r', p-1) = 1$. Again Bob finds $\overline{r'}$ which is the inverse element of r' from $r'\overline{r'} \equiv 1 \ (mod\ p-1)$.
4. Bob computes $Z' \equiv X^{r'y} = g^{r'xy} \ (mod\ p)$, again $v' \equiv g^{r'} \ (mod\ p)$ and sends (Z', v') to Alice.
5. Alice computes $Z \equiv Z'^{r} = g^{rr'xy} \ (mod\ p)$ and again $v \equiv X^r \cdot v'^x = g^{x(r+r')} \ (mod\ p)$ and sends (Z, v) to Bob.
6. Bob computes

$$K' \equiv Z^{\overline{r'}} = g^{r(r'\overline{r'})xy} = g^{rxy} \ (mod\ p).$$

and

$$K' \equiv v^y \cdot X^{-r'y} = g^{rxy+r'xy} \cdot g^{-r'xy} \ = g^{rxy} \ (mod\ p).$$

to verify if $Z^{\overline{r'}} = v^y \cdot X^{-r'y}$. If the above equation stands up, the generated key is accepted and it means that Bob can be sure that he has a right key. This method is illustrated in the following diagram :

$$Alice \quad \xleftarrow{Z' \equiv X^{r'y} = g^{r'xy} \ (mod\ p), \quad v' \equiv g^{r'} \ (mod\ p)} \quad \xrightarrow{Z \equiv Z'^{r} \ (mod\ p), \quad v \equiv X^r \cdot v'^x \ (mod\ p)} \quad Bob$$

As noticed the direction of arrows is inverse, it means that when Bob wants to decipher a message enciphered by Alice sends her a value like $(Z',\ v')$ and gets $(Z,\ v)$. As we will explaint next no one except Bob can decrypt a ciphertext sent by Alice.

4.2 Properties of the Protocol

As we discussed before there is no protection against an active adversary in the man-in-the-middle attack. But about this protocol the advantages are as following :

- At least Alice assrures that she has the right key.
- Bob can verify whether he has gotten the right key or not.

- This scheme is non-deterministic, since it uses randomization in the encryption process.

The other property of this scheme is that an active adversary cannot fool Alice into accepting an " invalid " key as valid, So he tries to fool Bob. But Bob can verify whether he has a valid key or not.

4.3 Security Aspects : Attacks

The main objective of this section is to highlight the delicate nature of the protocol discussed above. The security issues will be examined from every possible aspect. Attacks against *Diffie-Hellman protocol* comes in different flavors[17] that generally are classified as following :

- **Denial of service Attack:** Here, the attacker will try to stop Alice and Bob from successfully carrying out the protocol. The attacker can accomplish this in many ways, for example by deleting the messages that Alice and Bob send to each other, or by overwhelming the parties with unnecessary computation or communication. The plausibility of this attack depends on what assumption we make about the adversary. For example, if the adversary can remove and replace any message from the public communication file, the denial of service attack is impossible to prevent.
- **Outsider Attack:** In this attack, the attacker tries to disrup the protocol (by for example adding, removing, replaying messages) so that he gets some interesting knowledge (i.e. information he could not have gotten by just looking at the public values). An example of this attack comes with the man-in-the-middle attack.
- **Insider Attack:** It is possible that the recipient of the message in this protocol creates a breakable protocol run on purpose in order to try to gain knowledge of the participants about the secret key of his peer. This is an important attack if one of the participants holds a static secret key that is used in many key agreement protocol runs. This attack is prevented in this protocol when Bob wants to generate the shared key. Even if, Bob uses a static secret key, he has to delete it by computing $Z^{\overline{r'}}$ to generate the shared key.

4.3.1 Man-in-the-middle Attack

(Case 1)

In this protocol there is no way to fool Alice because Alice herself is the generator of the shared key. So, the only way for an active adversary in the man-in-the-middle attack is to fool Bob that it is considerd as following:

1. When Bob sends his $(Z',\ v')$ to Alice, an active adversary may know $(Z',\ v')$ and he substitutes $(Z'',\ v'')$ for $(Z',\ v')$ and sends it to Alice.

$$Z'' \equiv X^{r''w} = g^{r''xw}, \quad v'' \equiv g^{r''}\ (mod\ p).$$

2. Alice computes

$$Z \equiv Z''^{r} = g^{rr''xw}, \quad v \equiv X^{r} \cdot v''^{x} = g^{x(r+r'')} \ (mod\ p).$$

and sends it to Bob.

3. An active adversary computes

$$K'' \equiv Z^{\overline{r''}} = g^{rxw} \ (mod\ p).$$

or

$$K'' \equiv v^{w} \cdot X^{-r''w} = g^{rxw+r''xw} \cdot g^{-r''xw} = g^{rxw} \ (mod\ p).$$

which is different from real value $K \equiv g^{rxy}$. Thus, he cannot generate the right key. Meanwhile, Bob computes,

$$K' \equiv Z^{\overline{r'}} = g^{r\overline{r'}r''xw} \ (mod\ p),$$

or

$$K' \equiv v^{y} \cdot X^{-r'y} = g^{rxy+r''xy} \cdot g^{-r'xy} \ (mod\ p).$$

as the value of the keys is different Bob realizes that he does not have a valid key.

(Case 2)

This senario happens when Alice sends $(Z,\ v)$ to Bob.

1. When Alice sends her $(Z,\ v)$ to Bob, an active adversary can know it and he substitiutes $(Z'',\ v'')$ for $(Z,\ v)$.

$$Z'' \equiv Z'^{r''} = g^{r'r''xy}, \quad v'' \equiv W^{r''} \cdot v'^{w} = g^{w(r'+r'')} (mod\ p),$$

where $W \equiv g^{w}\ (mod\ p)$.

2. Bob computes

$$K' \equiv (Z'')^{\overline{r'}} = g^{r''xy} \ (mod\ p),$$

or

$$K' \equiv v''^{y} \cdot X^{-r'y} = g^{r''wy+r'wy} \cdot g^{-r'xy} \ (mod\ p).$$

as the value of the keys is different Bob realizes that he does not have a valid key.

Even with this case an adversary cannot extract the secret information from the exchanged values. Moreover, he cannot fool Bob into accepting an"invalid " key as valid one. To do this he needs extra information like the secret value y. There are various cases that an active adversary tries to extract the secret values of Alice and Bob, or to fool Bob into accepting an invalid key as valid by substitution, unfortunately, he fails, because the way that Bob generates a key depends on his secret key y.

4.3.2 Attacks based on Number Theory

The previous man-in-the-middle attack, although it completely breaks the protocol, requires the attacker to be very powerful. The followings are some cases that may occur from mathematical point of view which are related to number theory.

Degenerate Message Attack

There are degenerate cases in which the protocol does not work. For example when g^x or g^y equals one, the transferred data and the shared key becomes 1. Since the communication channel is public anybody can detect this anomaly. Fortunaltely, this situation is impossible because both x and y are chosen from $\{2, \ldots, p-2\}$.[1] Note that, if the computer program does not realize that g^x, g^y and g^{xy} cannot equal 1, the protocol is vulnearble. The same argument holds for values of the form $g^{\alpha\cdot(p-1)\cdot x}$ or $g^{\alpha\cdot(p-1)\cdot y}$, where $\alpha \geq 1$. So it is safe practice to always verify that g^x and g^y are positive integers smaller than $p-1$ and greate than 1.

There is another case that an adversary can detect the shared key when $r' = 1$. In this case $Z' \equiv X^{r'y} = g^{xy}$, and Alice computes $Z \equiv Z'^r = g^{rxy} = K$ which is the share key. Note that the same senario happens when $r = 1$ that in this case the generated key will be $Z \equiv Z'^r = g^{r'xy}$, and the attacker can get some information about the x and naturally he can find the key. Of course, when both $r = 1$ and $r' = 1$, the shared key will be $g^{xy} \equiv K$ which will be seen on the public channel. Fortunaltely, this situation is impossible because both r and r' are chosen from $\{2, \ldots, p-2\}$.

Attacks Based on Composite Order Subgroups

In this attack, the attacker can exploit subgroups that do not have large prime order[7]. This is best illustrated by an example. Suppose Alice and Bob choose a prime $p = 2q+1$, where q is prime, and the gererator g of order $p-1 = 2q$. The attacker can intercept the messages g^x and g^y and exponentiate them by q (he will replace g^x by g^{xq} and g^y by g^{yq}.) In Diffie-Hellman protocol the secret key will be g^{xyq} which allows the attacker to find this value by exhaustive search[2].

Let us consider an insider attack by Bob using the attack explained above. As discussed before, even if Bob uses q instead of r to know the secret value of Alice, it is impossible for him to do this becuase after receiving $Z \equiv Z'^r = g^{rqxy}$ he has to compute $Z^{\overline{q}} \equiv g^{rq\overline{q}xy} = g^{rxy}$, which means that the generated key does not depend on Bob's session key. In spite of that, the protocol protects Alice's secret key in an insider attack, there is a lesson to be learned from this attack that we should choose g that generates a large prime order subgroup or at the very least to make sure that composite order subgroups are not vulnerable. Moreover, to choose a prime p such that $p-1$ contains large factors.

[1] if g is a generator of $\mathbb{Z}_p^*$, $g^z = 1 \ (mod\ p)$ iff $z = 0 \ (mod\ p-1)$.

[2] the subgroup generated by $g^{(p-1)/2}$ is $\{g^{(p-1)/2} = p-1,\ (g^{(p-1)/2})^2 = 1\}$.

4.3.3 Other types of Attacks

In the previous section attacks related to the mathematical structure were considered. In this section issues related to other types of attacks are addressed about this research.

- **Message Redirection:** It is possible for an adversary in the man-in-the-middle-attack to intercept and send it to someone other than the intended recipient. Note that, this attack can be used with ephemeral Diffie-Hellman not with the protocol in this research becuase an attacker cannot intercept any key with Alice or Bob. Even if, the attacker sends the messages to someone else, it is impossible for him or her to decipher it becuase if it had been possible the attacker himself had done it.

- **Ciphertext only Attack:** In this attack, the cryptanalyst is given $c_1 = E_1(m_1)$, $c_2 = E_2(m_2), \ldots c_i = E_i(m_i)$, the enciphering of i distinct unknown cleartext messages under the same unknown key. He is to infer the key K or, lacking this ability , as many among $m_1, m_2, \ldots, m_i$ as possible.
  The protocol proposed in this research is secure against this attack becuase a random number r is used in the shared key. As the random number refreshes the key for every message it will be unachievable for a cryptanalyst to find the key. Note that, even if he can find the shared key, it will be useless for the next time.

- **Known plaintext Attack:** The cryptanalyst is given $c_1, c_2, \ldots, c_i$ as above, but also the corresponding $m_1, m_2, \ldots, m_i$. He is to infer K or, lacking this ability, he is to infer m_{i+1} from some new ciphertext $c_{i+1} = E_{i+1}(m_{i+1})$.
  The same senario mentioned above happens to cryptanalysts becuase the random number used in the shared key refreshes it. Moreover, to do $c_{i+1} = E_{i+1}(m_{i+1})$, he needs some information about x and y which are secret and nobody excepts the owners knows them.

- **Chosen plaintext Attack:** The cryptanalyst gets to choose plaintext messages $m_1, m_2, \ldots, m_i$ and he is given the corresponding $c_1 = E_1(m_1), c_2 = E_2(m_2), \ldots, c_i = E_i(m_i)$. He is to infer K or lacking this ability, he is to infer m_{i+1} from some new ciphertext $c_{i+1} = E_{i+1}(m_{i+1})$.
  In this attack the cryptanalyst confront with the same problem mentioned above. If the enciphering key were the same for each protocol run it might be possible for the cryptanalyst to know some information about the shared key. But, unfortunately, the shared key is refreshed for every protocol run.

- **Chosen ciphertext attack:** The cryptanalyst gets to choose ciphertext messages $c_1, c_2, \ldots, c_i$, and he is given the corresponding $m_1 = D_1(c_1), m_2 = D_2(c_2), \ldots, m_i = D_i(c_i)$, provided they exist. He is to infer K or any efficient algotithm for computing D_i or, lacking this ability, he is to infer m_{i+1} from some new ciphertexts $c_{i+1} = E_{i+1}(m_{i+1})$.
  This attack is similary to chosen plaintext attack with considerting that $D_i = \overline{K}$, and $E_i = K$, where $K \cdot \overline{K} = 1 \ (mod\ p-1)$. Thus, the same problem confronts cryptanalyst.

4.3.4 Security Considerations

In this section, some recommendations are given which should be taken into account when implementing Diffie-Hellman based on protocols. Most of these recommendation are based on the attackes discussed before.

- **Parameter Authentication:** As a general principle, all parameters used in a cryptographic protocol should be authenticated. For example, suppose the paticipants do not authenticate their choice of parameters, an attacker might be able to fool them into using weak parameters. This type of attack can be very subtle and can even be missed by top cryptogphers and security experts [17].

- **Deleting the secret exponents:** It is important to delete the secret exponents, to guard against memory beign written to disk and prevent unwanted access to these values.

- **Key Freshness and Perfect Forward Secrecy:** In many situations the shared secret key should be changed frequently, like the protocol proposed in this research. Here are the main reasons why we might want to obtain new shared secret keys.

  1. *Reduce Exposure* The propability that a given key is compromised is lower if it is not used often.
  2. *Forward Secrecy* If old encryption keys are deleted, encrypted messages can no longer be decrypted. Hence, a third party cannot mount a ciphertext only (chosen plaintext attack, and chosen ciphertext) attack.

- **Key Independence:** As a general principle, we always want to have independent keys. Precisely, obtaining one secret key should not help an attacker uncover other keys. This property is called *known key security.*

- **Protocol Math**

  1. Spot Unconventional Messages
     - Make sure that g^x, g^y and g^{xy} do not equal 1.
     - Make sure that g^x and g^y are less than $p-1$ and greater than 1.
     - Choose x, y, from the set $\{2, \ldots, p-2\}$.
  2. Be careful About g's Order
     - The prime factor decomposition of the order of g should not be composed entirely of small primes.
     - The subgroup generated by g should not have small order subgroup. If at all possible, construct and use a generator that has a large prime order.

- **Efficiency** Ideally, the generator g should be as small as possible in order to reduce the cost of modular exponentiation.

Chapter 5

Conclusions

The results of this research are as following:

1. The generated key is secure against the man in the middle attack.
2. The protocol is based on intractability of Diffie-Hellman problem.
3. It is capable for implicit key authentication, and key confirmation, it provides assurance for the recipient whether he or she has computed the valid key.
4. Using random numbers for session keys it is non-deterministic.
5. It is secure against the attacks discussed in this research.
6. If the considerations are maintained it can be protected against the attacks based on number theory as well.

Diffie-Hellman key exchange algorithm, is based on the assumption that discrete logarithms are hard to compute. This intractability hypothesis is also the foundation for the presumed security of a variety of other public key schemes. While there have been substantial advances in discrete log algorithms in the last two decades, in general the discrete log still appears to be hard. Unfortunately no proofs of hardness are available in this area, so it is necessary to rely on experience and intuition in judging what parameters to use for cryptosystems.

As we disscussed the proposed protocol is secure and its security is based on Diffie-Hellman problem and the complexity is the same. This protocol is non-deterministic, since it uses randomization in the encryption process. The other property of this protocol is verification that can be done easily by the recipient of the message. This protocol has been extended from the methods discussed in chapter 3.

One open problem with this protocols is that it is still unknown whether it can be generalized and extended to be used among 3 or more users. This protocol can be used as a one-pass protocol as well. In this situation it will not be secure against most of the attackes considered in chapter 4.

The protocol proposed in this research possesses many desirable security attributes. It is hoped that the protocol, or appropriate modifications of it, can, under plausible assumption, be proven secure in the model of distributed computing.

Bibliography

[1] Neal Koblitz,"*A Course in Number Theory and Cryptogaphy* ", Second Edition, Springer, (1994).

[2] Neal Koblitz,"*Algebraic Aspect of Cryptography* ", Springer, (1997).

[3] Bruce Schneier,"*APPLIED CRYPTOGRAPHY* ", Second Edition, John Wiley & Sons Inc., (1996).

[4] G. H. Hardy, E. M. Wright,"*An Introduction to the Theory of Numbers* ", Fifth Edition, Oxford Science Publication, (1992).

[5] Jennifer Seberry, Josef Pieprzyk,"*CRYPTOGRAPHY An Introduction to Computer Security* ", Prentice Hall, (1988).

[6] Douglas R. Stinson,"*CRYPTOGRAPHY Theory and Practice* ", CRC Press, (1995).

[7] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone,"*Handbook of Applied Cryptography* ", CRC Press, (1996).

[8] 池野信一・小山謙二,"現代暗号理論 ", 電子情報通信学会, (1997).

[9] 澤田 秀樹,"暗号理論と代数学 ", 海文堂出版, (2000).

[10] 山本芳彦,"数論入門１ ", 岩波書店, (1996).

[11] 今井 秀樹,"暗号のおはなし ", 日本規格協会, (1996).

[12] 遠山 啓,"初等整数論 ", 日評数学選書 : 日本評論社, (1992).

[13] David M. Bressoud,"*Factorization and Primality Testing* ", Spring-Verlag, (1996).

[14] Whitfield Diffie and Martin E. Hellman,"*New Directions in Cryptography* " Invited Paper, (1976).

[15] Ronal Cramer, Victor Shoup,"*A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack* ", (1998).

[16] D. R. Stinson,"*Finding Primitive Elements in* $\mathbb{Z}_p$ "

[17] Jean-Fransico Raymond and Anton Stiglic,"*Security Issues in the Diffie-Hellman Key Agreement Protocol* " Zero-Knowledge System Inc., (2000).

[18] Simon Blake-Wilson and Alfred Menezes,"*Authenticated Diffie-Hellman Key Agreement Protocols* " , (1998).

[19] Tsutomu Matsumoto, Youichi Takashima, and Hideki Imai,"*On seeking Smart Public-Key-Distribution Systems* " The Transactions of the IECE of Japan, Vol. E **69**, No. 2, (1986).

[20] D. R. Stinson,"*The Pohlig-Hellman Algorithm* "

[21] Simon Blake-Wilson, Don Johnson, Alfred Menezes,"*Key Agreement Protocols and their Security Analysis* " (1997).

[22] Louis C. Guillou, Jean,"*Advances in cryptography, CRYPTO 95 : Workshop on the Theory and Application of Cryptographic Techniques* ", Sain-Malo, France, (1995).

[23] S. Goldwasser,"*Advances in cryptography, EUROCRYPT 88 : Lecture notes in computer science ; 403* ", Spring-Verlag, (1990)

[24] Gilles Brassard,"*Modern cryptography, a tutorial : Lecture notes in computer science ; 325* ", Spring-Verlag, (1988).

[25] Th. Beth, M. Frisch, G. J. Simmons,"*Pblic-key cryptography : state of the art and futrure directions* ", E.I.S.S. workshop, Oberwolfach, Germany (1991).

[26] Andrew Odlyzko,"*Discrete logarithms: The past and the future* ", AT&T Labs - Research, (1999).