UNIT 7 PUBLIC KEY ENCRYPTION

Structure	Page No.
7.1 Introduction	5
Objectives	
7.2 Public Key Cryptosystems	6
7.3 The RSA Cryptosystem	7
7.4 Diffie-Hellman Key Exchange and the ElGamal Cryptosystem	13
Diffie-Hellman key exchange	
ElGamal Cryptosystem	
7.5 Integer Factorisation and Discrete Logarithms	18
Computing Discrete Logarithms	
7.6 Summary	31
7.7 Solutions/Answers	31

7.1 INTRODUCTION

In the previous block, we discussed some symmetric key cryptosystems where we can obtain the decryption key from the encryption key. Because of this reason, the key has to be kept secret and the parties that want to communicate securely have to find a way of exchanging the keys. This was not a problem in the earlier times when the cryptography was not used widely and the users were very few.

However, things changed with invention of more communication tools and the digital computer. Many commercial organisations started using cryptography and they faced the problem of distributing keys. The organisations exchanged keys through trusted couriers or meeting in person. In their path breaking paper [5], Diffie and Hellman proposed the framework for a new cryptosystem called **public key cryptosystem**. However, they did not give any concrete example. In public key cryptosystem, the encryption key can be made public without compromising the security. So, people can send messages to each other without exchanging keys before.

In Sec. 2, we will introduce you to public key cryptosystems. In Sec. 3, we will discuss RSA public key cryptosystem. In Sec. 4, we will discuss the ElGamal public key cryptosystem. In the same paper where they discussed the public key crypto system, Diffie and Hellman also gave a method of exchanging keys without meeting in person or using trusted couriers. We will discuss this also in Sec. 4. The security of the RSA cryptosystem is based on the fact that integer factorisation is a 'hard' problem. Similarly, the security of the ElGamal cryptosystem is based on the fact that discrete logarithm problems is a 'hard' problem. we will discuss the integer factorisation problem and discrete logarithms problem in Sec. 5.

Objectives

After studying this unit, you should be able to

- explain what is a one way function;
- explain what is a cryptographic protocol;
- explain the RSA algorithm;
- explain the ElGamal algorithm;
- explain Fermat factorisation algorithm, $p-1$ algorithm and the quadratic sieve algorithm; and

- explain Pohling-Hellman algorithm, Baby Step-Giant Step alogrithm and the index calculus algorithm for discrete logarithms.

7.2 PUBLIC KEY CRYPTOSYSTEMS

In Block 2, we discussed symmetric key cryptosystems like DES, AES and RC4. The drawback in these systems is that the communicating parties have to exchange the keys securely before they start their communication. Another issue is the huge storage required for storing all the encryption and decryption keys. Also, until the parties exchange the keys, it is not possible for the parties to communicate securely. In commercial activities, organisations need to communicate securely with other organisations or persons with whom they have not had any previous transaction. In such cases, commercial organisations may lose business opportunities if they wait till they decide upon the keys.

To overcome this difficulty, in [5], Diffie and Hellman described a new concept, the public key cryptosystem. However, they didn't propose any specific cryptosytem based on this. As we will see later, specific systems that were based on this concept were invented by others. In the same paper they also proposed a method by which two parties that want to communicate with each other can exchange keys securely without meeting in person.

Note that, in the case of symmetric key cryptosystems, we assume that it is not possible break the cryptosystem by exhaustive key search, i.e. it will require impossibly large memory and time to try out all the keys. As we will see in the definition that follows, in the public key cryptosystems, we assume that it is 'computationally infeasible' to find the decryption key from the encryption key.

Before we formally define a public key cryptosystem, we need the following definition:

Definition 1: A task is **computationally infeasible** if either the memory required or the time taken for carrying out the task is finite, but impossibly large.

We begin by formally defining a public key cryptosystem.

Definition 2: A **public key cryptosystem** is a pair of families $\{E_k\}_{k\in\mathscr{K}}$ and $\{D_k\}_{k\in\mathscr{K}}$ of functions satisfying

You may like to refer to pages 51 and 52 of Block 1 to recall what $\mathscr{K}$, $\mathscr{P}$ and $\mathscr{C}$ are.

$$E_k : \mathscr{P} \longrightarrow \mathscr{C} \tag{1}$$

and

$$D_k : \mathscr{C} \longrightarrow \mathscr{P} \tag{2}$$

such that:

1) For every $k \in \mathscr{K}$, E_k is the inverse of D_k.
2) For every $k \in \mathscr{K}$ and $p \in \mathscr{P}$, $c \in \mathscr{C}$, $E_k(p)$ and $D_k(c)$ are easy to compute.
3) For every $k \in \mathscr{K}$, it is feasible to compute inverse pairs E_k and D_k from k.
4) For almost every $k \in \mathscr{K}$, it is computationally infeasible to derive D_k from E_k without knowing k.

Property 4 of Definition 2 allows us to disclose the enciphering function E_k without compromising security. The cryptosystem has two parts, a family of enciphering transformations and a familiy of deciphering transformations. It is infeasible to find the deciphering transformation from the enciphering transformation.

Property 3 of Definition 2 guarantees that we can compute the corresponding pairs of inverse transformations when there is no constraint on what the enciphering transformation or deciphering transformation should be.

Definition 3: A bijective function $f\colon A \longrightarrow B$ is a **one way** function if we can compute $f(a)$ easily for any $a \in A$, but, given $b \in B$, it is computationally infeasible to compute its inverse image $f^{-1}(b)$.

After the paper [5] was published, many people provided examples of one way functions and designed cryptosystems based on them. We will discuss some of them now.

In [14], Rivest, Shamir and Adleman described the RSA cryptosystem. This is one of the most popular systems for public key encryption and is widely used. Let us now discuss this encryption algorithm.

7.3 THE RSA CRYPTOSYSTEM

Suppose you are given two 300 digit prime numbers and asked to multiply them, you can multiply them quite quickly on a computer. But, if you are given a product of two carefully chosen 300 digit primes and asked to find what the primes are, it may take millions of years to find it using the knowledge and the computing power available now. This was the idea behind the design of the RSA cryptosystem that we will study in this section.

Of course, with improvements in computational power, the number of digits required to make factorisation computationally infeasible will increase.

Suppose Bob wants to use the RSA cryptosystem to receive messages. He does the following:

1) Selects two large primes p and q and multiplies them to get $n = pq$.

2) Selects e such that

$$(e, (p-1)(q-1)) = 1. \tag{3}$$

Recall that (a, b) denotes the greatest common divisor of a and b.

Note that, $(p-1)(q-1) = \phi(n)$ where $\phi(n)$ is the Euler's ϕ function(see Block 1 of this course and Block 1 of MMT-003.). He can then find integers d, d' using extended euclidean algorithm(see Unit 2 of Block 1)such that

$$de + d'(p-1)(q-1) = 1, \text{ i.e. } de \equiv 1 \pmod{(q-1)(p-1)}. \tag{4}$$

We call n, the **modulus of encryption**, e, **the encryption exponent** and d, the **decryption exponent**.

3) Publishes the values of n and e, but keeps p, q and d a secret.

Now, if Alice wants to send a message to Bob, she looks up the values of n and e. She represents the message by a number, say $\mathscr{M}$, $0 \leq \mathscr{M} \leq n-1$, and calculates

$$c \equiv \mathscr{M}^e \pmod{n}$$

and sends the value c to Bob. If the message is long, she breaks up the message into blocks so that each block can be represented by a number between 0 and $n-1$. We will discuss some methods for converting text into numbers in the practical guide. Here, we will assume that we have converted text to a number using any of the methods.

Bob decrypts the message by raising c to the power d modulo n. Of course, both Bob and Alice can calculate the powers using repeated squaring algorithm we discussed in

Block 1. Before we prove that the encryption and decryption work correctly, let us look at some examples.

Example 1: Let us take p = 13, q = 19. Then, n = 247 and $\phi(n) = 12 \cdot 18 = 216$. Let us take e = 7. Note that, this is a valid choice since $(7, 216) = 1$. Using extended euclidean algorithm, we get $7 \cdot 31 - 216 = 1$, so d = 31.

Suppose the message is $\mathcal{M} = 115$. We have,

$$115^7 \equiv 210 \pmod{247}.$$

We can calculate $115^7 \pmod{247}$ using repeated square algorithm in page 34 of Block 1 with the help of a calculator. Here are the values at the end of each iteration, so that you can check your calculations.

At the end of iteration 1, m = 3 P = 115 b = 134
At the end of iteration 2, m = 1 P = 96 b = 172
At the end of iteration 3, m = 0 P = 210 b = 191

Here, we have calculated all the values modulo 247. We have also used the same notation as in Block 1. So, c, the encrypted text is 210.

Again, to decrypt, we raise 210 to the power 31 modulo 247. Here are the calculations using repeated squaring method:

At the end of iteration 1, m = 15 P = 210 b = 134
At the end of iteration 2, m = 7 P = 229 b = 172
At the end of iteration 3, m = 3 P = 115 b = 191
At the end of iteration 4, m = 1 P = 229 b = 172
At the end of iteration 5, m = 0 P = 115 b = 191

So, $210^{31} \equiv 115 \pmod{247}$. So, the encryption and decryption functions work correctly.

* * *

Here are some exercises for you to check your understanding.

E1) Take p = 7, q = 11. Find $\phi(n)$. Check that e = 6 is a proper encryption exponent and if it is not, find a suitable encryption and decryption exponent. Encrypt the message $\mathcal{M} = 40$ and check your decryption exponent by checking that c decrypts correctly.

E2) Let p = 17, q = 23 and e = 71. If the ciphertext is 171, find the plaintext.

Let us now see why the RSA algorithm is correct. Suppose $\mathcal{M}$ is the plain text and c is the cipher text.

We have

$$c^d \equiv (\mathcal{M}^e)^d \equiv \mathcal{M}^{ed} \pmod{n} \tag{5}$$

We have $\mathcal{M}^{ed} = \mathcal{M}^{1+k\phi(n)}$ for some integer k since $ed \equiv 1 \pmod{\phi(n)}$. So, if $(\mathcal{M}, n) = 1$, then $\mathcal{M}^{ed} \equiv \mathcal{M} \pmod{n}$ since $\mathcal{M}^{\phi(n)} \equiv 1 \pmod{n}$ by Euler's theorem.

If $(\mathcal{M}, n) \neq 1$, we have $p \mid \mathcal{M}$, $q \nmid \mathcal{M}$, or $q \mid \mathcal{M}$, $p \nmid \mathcal{M}$ or both p and q divide $\mathcal{M}$. Let us first consider the case $p \mid \mathcal{M}$, $q \nmid \mathcal{M}$. Then, $\mathcal{M} \equiv 0 \pmod{p}$, so $\mathcal{M}^{ed} \equiv 0 \equiv \mathcal{M} \pmod{p}$.

Since $(\mathcal{M}, q) = 1$, $\mathcal{M}^{q-1} \equiv 1 \pmod{q}$ by Fermat's theorem. So,

$$\mathcal{M}^{1+k(p-1)(q-1)} \equiv \mathcal{M} \cdot \mathcal{M}^{k(p-1)(q-1)} \equiv \mathcal{M} \cdot \left(\mathcal{M}^{q-1}\right)^{k(p-1)} \equiv \mathcal{M} \pmod{q}.$$

In other words, $p \mid \left(\mathcal{M}^{ed} - \mathcal{M}\right)$, and $q \mid \left(\mathcal{M}^{ed} - \mathcal{M}\right)$. Therefore, since $(p, q) = 1$, $pq \mid \left(\mathcal{M}^{ed} - \mathcal{M}\right)$, i.e. $\mathcal{M}^{ed} \equiv \mathcal{M} \pmod{n}$.

We can dispose the other case, $q \mid \mathcal{M}$, $p \nmid \mathcal{M}$, similarly. If $\mathcal{M}$ is divisible by both p and q, then $\mathcal{M} \equiv 0 \pmod{n}$, So, $\mathcal{M}^{ed} \equiv 0 \equiv \mathcal{M} \pmod{n}$.

Let us now verify that the RSA cryptosystem has the properties mentioned in Definition 2. Let us take

$$\mathcal{K} = \{(e, p, q) \in \mathbf{N} \times \mathbf{N} \times \mathbf{N} \mid (e, (p-1)(q-1)) = 1\}.$$

For each $k = (e, p, q) \in \mathcal{K}$, we have $E_k : \mathcal{P} \longrightarrow \mathcal{C}$ given by $E_k(\mathcal{M}) \equiv \mathcal{M}^e \pmod{n}$ and $D_k(c) \equiv c^d \pmod{n}$ where d is uniquely determined $\bmod (p-1)(q-1)$. So, condition 1) of Definition 2 is satisfied. We can easily compute these powers by repeated squaring algorithm. So, 2) of Definition 2 is also satisfied.

Of course, once we know e, p and q, we can easily calculate d. So, condition 3) is also satisfied.

Regarding 4), it is believed by experts that only way of finding d from n and e is to factorise n, find $\phi(n)$ and use it to find d from e and $\phi(n)$ by extended euclidean algorithm. Of course, we cannot prove that factorising n is the only way to find $\phi(n)$, though experts believe this is so. DeLaurentis, [4], and Miller,[8], show that computing an RSA private key (p, q, d) from its public key (n, e) is as difficult as factoring.

Remark 1: Note that, if we know n and $\phi(n)$, we can recover p and q as follows: We have

$$\phi(n) = pq - p - q + 1 = n - (p+q) + 1$$

Since we know n, we can find $p+q$ from the above equation. Since we know $pq = n$ and $p+q$, we can find p and q by factoring the quadratic equation

$$x^2 - (p+q)x + pq = 0.$$

For example, if we know that $n = 29591$ and $\phi(n) = 29232$. Then,

$$\phi(n) = 29232 = n - (p+q) + 1 = 29591 - (p+q) + 1$$
$$\therefore p + q = 29591 + 1 - 29232 = 360$$

So, p and q are the roots of the equation $x^2 - 360x + 29591$. Solving, we get the roots 127 and 233. So, $29591 = 127 \cdot 233$.

Here is an exercise to test your understanding of the above remark.

E3) You are given that $n = 31309$ and $\phi(n) = 30940$. Find the factors of p and q.

What we have described is the bare bones version of the RSA algorithm. However, when we implement it, we have to take many precautions. Let us now discuss some of them now.

We mentioned that the only way of finding the decryption exponent seems to be factoring n. It is easy to factorise n if the primes are not properly chosen. If p and q are

close to each other so that $|p-q|$ is small, then we can factorise n using Fermat's method that we will discuss in the section on integer factorisation.

Also, if $p-1$ has small prime factors, we can factorise n using Pollard's $p-1$ algorithm; we will discuss this algorithm in Sec. 5. If $p+1$ has small prime factors, then we can factor n using William's $p+1$ algorithm. See [17] for a discussion of this algorithm. For these reasons, we have the notion of strong prime. A prime p is called a **strong prime** if

1) $p-1$ has a large prime factor, say r.

2) $p+1$ has a large prime factor.

3) $r-1$ has a large prime factor.

However if we choose a large prime at random, then we expect $p-1$ and $p+1$ to have large prime factors. Even if we choose strong primes, we can factor n using the number field sieve if the primes are not large enough. So, it makes better sense to choose a large prime factor at random since this affords protection both against $p-1$ and $p+1$ factoring algorithms as well as the modern algorithms like the number field sieve. See [12] for a discussion on this.

Let us now discuss how to generate a random prime. Suppose we want to generate a k-bit prime m. We set the first and last bit of m to 1 and use a pseudo random bit generator to generate the remaining $k-2$ bits. We choose random bits that are uniformly distributed. We first check by trial division if m is a prime by dividing m by all primes up to a certain limit, usually, 10^6. We then apply the Rabin-Miller test three times. If the test returns NO, then probability of m being composite is less than $\frac{1}{2^{80}}$. We then consider m to be a prime. If m turns out to be composite, the receiver of messages(Bob) will detect it because the decryption will not work properly. In this case, Alice can generate a new set of primes and start afresh. Note that, this has very negligible probability.

See pages 41 and 42 of Block 1 for a discussion of Rabin-Miller test.

The decryption exponent should be sufficiently large so that it cannot be found using brute force. Also, in [16], Wiener proves that if $d < \frac{n^{1/4}}{3}$, we can calculate d from n and e.

Interestingly, according to a result of Miller, if we know d and n, we can factorise n using a probabilistic algorithm. We will discuss this method later in the section on integer factorisation.

Another issue regarding the security of the RSA is the choice of encrpytion exponent. Suppose there are three recipients, all of them with the same encrption exponent 3, but with different modulus n_i, $i = 1, 2, 3$. Then, we calculate $y_i = \mathscr{M}^3 \pmod{n_i}$ and send them to the recipients. Suppose two of them, say n_1 and n_2, are not coprime. Then, (n_1, n_2) is a non-trivial factor of n_1 and n_2 and any adversary can factorise both of them. So, we can always assume that n_i are pairwise coprime.

If Eve gets hold of the messages y_i, $1 \leq i \leq 3$, she can compute

$$\mathscr{M}^3 \bmod n_1 n_2 n_3$$

using chinese remainder theorem since n_i are pairwise coprime. Since $m < n_i$, $m^3 < n_1 n_2 n_3$. So, $\mathscr{M}^3 \bmod n_1 n_2 n_3$ is equal to $\mathscr{M}^3$ and the adversary can find $\mathscr{M}$ by taking the cube root of $\mathscr{M}^3 \bmod n_1 n_2 n_3$. Because of this, the encryption exponent should not be too small. Let us now look at an example.

Example 2: Suppose Alice encrypts a message $\mathscr{M}$ using exponent 3 and three different moduli, 799, 1159 and 1909. She gets three different values $593 \pmod{799}$,

$1122 \pmod{1159}$ and $1727 \pmod{1909}$. She then sends it to 3 different recipients. Suppose, Eve manges to get hold of all the 3 encrypted messages. She solves the congruences

$$x \equiv 593 \pmod{799} \quad x \equiv 1122 \pmod{1159} \quad x \equiv 1727 \pmod{1909}$$

using Chinese remainder theorem. We have $n_1 = 799$, $n_2 = 1159$, $n_3 = 1909$. So, $N = n_1 \cdot n_2 \cdot n_3 = 1767812269$, $N_1 = n_2 \cdot n_3 = 1159.1909 = 2212531$, $N_2 = 1525291$ and $N_3 = 799 \cdot 1159 = 926041$. We have $a_1 = 593$, $a_2 = 1122$ and $a_3 = 1727$. Recall that N_i' is such that $N_i N_i' \equiv 1 \pmod{n_i}$. We can find N_i' using extended euclidean algorithm. We have, $N_1' = 8$, $N_2' = 74$, $N_3' = 1768$. So,

See page 74 of Block 1, MMT-003 study guide for a discussion of Chinese Remainder Theroem.

$$\begin{aligned} x &= 593 \cdot 2212531 \cdot 8 + 1122 \cdot 1525291 \cdot 74 + 1727 \cdot 926041 \cdot 1768 \\ &= 2964652430988 \\ &\equiv 31255875 \pmod{1767812269} \end{aligned}$$

Taking cube root, she will get $\sqrt[3]{31255875} = 315$, thus recovering the message $m = 315$.

* * *

Try an exercise now to check your understanding of the above example.

E4) Eve knows that Alice has encrypted a message using the exponent 3 and 3 different moduli. The encrypted messages and the modulii are $78 \pmod{989}$, $254 \pmod{1273}$ and $66 \pmod{1513}$. Find the message.

Since the encryption key is known in RSA, we may succeed in a known plain text attack. Consider the following situation. Suppose Bob sends Alice e-mails periodically, informing about the date of a particular event. Suppose Eve gets hold of one of these messages and knows that the date of the event in the letter is in a specific format. To find the date, Eve has to just encrypt all the 365 dates in the format used by Bob and check if the encrypted text in Bob's message contains any of 365 possible texts.

We can overcome many of the problems we mentioned above by using the Optimally Assymetric Encryption Protocal(OAEP) discovered by Bellare and Rogaway, [1]. Our discussion follows [15].

In this method, Alice does the following: Suppose Bob's encryption exponent is e and the modulus is n. Further, suppose that the size of n is k bits. Alice and Bob fix three positive integers k_0, k_1 and k_2 such that $k_0 + k_1 + k_2 = k$. Alice's message should be k_2 bits long. (If length of the message is less than k_2, she can pad the message with zeros. If the length of the message is greater than k_2, she can always divide the message into chunks of size k_2. In this process, there may be at most one chunk with size less than k_2. She can pad this with zeros so that all the chunks have size k_2.) Let G be a pseudorandom generator that takes as input a string of size k_0 and outputs a string of size $k_1 + k_2$ bits and H be a function that takes as input a string of size $k_1 + k_2$ bits and outputs a string of size k_0 bits. We usually construct G and H from hash functions.

To encrypt the message Alice concatenates the message with k_1 zero bits and expands the message to $\mathscr{M}||0^{k_1}$ and this string has size $k_1 + k_2$. Next, she selects a random string r of length k_0 bits, uses it as the random seed for $G(r)$ and computes

$$x_1 = \mathscr{M}||0^{k_1} \oplus G(r), \qquad x_2 = r \oplus H(x_1) \tag{6}$$

Note that $G(r)$ and $\mathscr{M}||0^{k_1}$ have length $k_1 + k_2$. Also, r and $H(x_1)$ have length k_0. So, the XOR operations in Eqn. (6) are possible.

If $x_1||x_2$ is a binary number bigger than n, Alice choses another random string r and computes the new values of x_1 and x_2. If G(r) produces fairly random outputs, $x_1||x_2$ will be less than n in binary with a probability greater than half. After getting a string r with $x_1||x_2 < n$, Alice then encypts $x_1||x_2$ to get the ciphertext

$$E(\mathscr{M}) = (x_1||x_2)^e = c \pmod{n} \tag{7}$$

The process is summarised in Fig. 1. Note that, the inputs and outputs are in oval

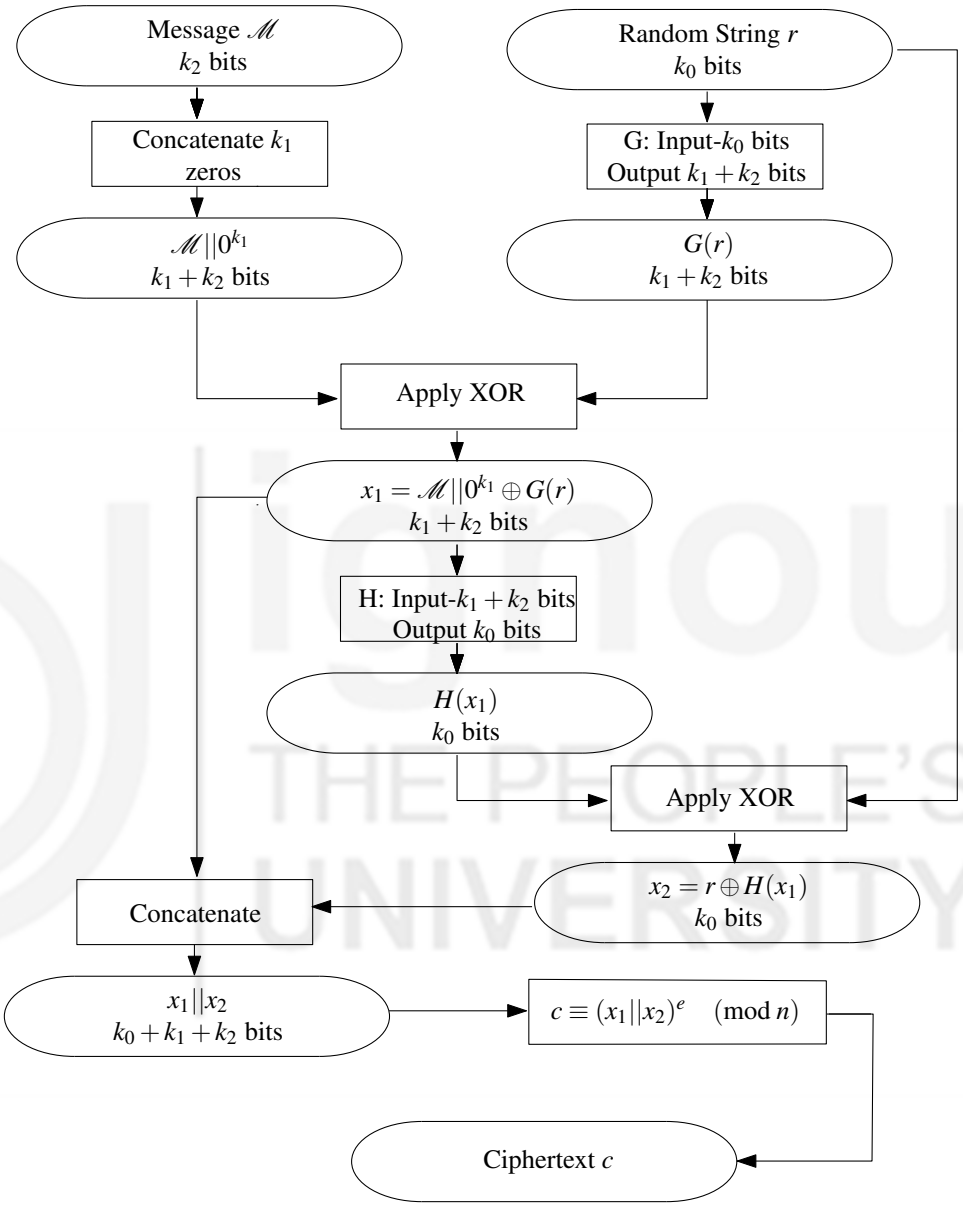

Fig. 1: OAEP encryption

boxes and the processes are in square boxes.

Bob decrypts the ciphertext by using his private key d and finding $c^d \pmod{n}$. He then writes the result in the form

$$c^d \pmod{n} = y_1||y_2 \tag{8}$$

where y_1 has k_1+k_2 bits and y_2 has k_0 bits. Note that $y_1 = x_1$ and $y_2 = x_2$ since

$$c^d \equiv ((x_1||x_2)^e)^d = (x_1||x_2).$$

Since $x_1 = \mathscr{M}||0^{k_1} \oplus G(r)$, Bob can find $\mathscr{M}||0^{k_1}$ if he can find r and hence G(r) because

$$y_1 \oplus G(r) = x_1 \oplus G(r) = \mathscr{M}||0^{k_1} \oplus G(r) \oplus G(r) = \mathscr{M}||0^{k_1}.$$

So, if Bob can find r, he can find $\mathscr{M}||0^{k_1}$, discard the last k_1 bits and recover the message.

Let us see how Bob can find r. Bob can recover the random string r by calculating

$$\begin{aligned} H(y_1) \oplus y_2 &= H(x_1) \oplus x_2 \\ &= H(x_1) \oplus r \oplus H(x_1) \text{ (from Eqn. (6))} \\ &= r. \end{aligned}$$

Note that we can use OAEP with any public key cryptosystem. We simply have to change the encryption and decryption functions. If we use this method, we cannot call the encryption operation a function on the plaintext space. This is because the same plain text may be encrypted differently at different times depending on the choice of the random string chosen. However, given the ciphertext we can always recover the plaintext provided we know the decryption exponent.

We have merely touched upon the various attacks on the RSA cryptosystem. For a more detailed discussion of other methods of attacking the RSA cryptosystem see [13], [15].

In the next section, we will discuss the ElGamal cryptosystem. This system is based on discrete logarithms. Before you read the next section, you may find it useful go through the discussion on discrete logarithms on pages 21, 22 and 23 of Block 1.

7.4 DIFFIE-HELLMAN KEY EXCHANGE AND THE ELGAMAL CRYPTOSYSTEM

In this section, we will discuss the ElGamal cryptosystem which is based on difficulty of finding discrete logarithms. This system was discovered by ElGamal in 1985. Let us start by recalling the definition and the basic properties of discrete logarithms over a finite field.

Definition 4: Let $\mathbf{F}_q$ be a finite field with q elements. For an element $\gamma \in \mathbf{F}_q^*$, suppose $\gamma = \alpha^k$ with $0 \leq k < q-1$. Then, we call k the **logarithm** of γ to the base α. We write $\log_\alpha \gamma = k$. If $0 \leq k < q-1$, then α^k is the **antilogarithm** of k to the base α. We write $\text{alog}_\alpha k = \gamma$.

Note that $\log_\gamma \colon \mathbf{F}_q^* \longrightarrow \{0,1,\ldots,q-2\}$ and $\text{alog}_\gamma \colon \{0,1,\ldots,q-2\} \longrightarrow \mathbf{F}_q^*$ are inverse functions of each other. Further, they satisfy the relations

$$\log_\gamma(\alpha\beta) \equiv \log_\gamma(\alpha) + \log_\gamma(\beta) \pmod{q-1} \tag{9}$$

$$\log_\gamma(\alpha\beta^{-1}) \equiv \log_\gamma(\alpha) - \log_\gamma(\beta) \pmod{q-1} \tag{10}$$

In the first block, we have already seen examples of discrete logarithms in general finite fields. Let us now look at an example over a finite field of the type $\mathbf{F}_p$, where p is a prime.

Example 3: Consider $\mathbf{Z}_{23}$ a finite field with 23 elements. Then, 5 is a primitive root for $\mathbf{Z}_{23}$. In Table 1 we have computed all the powers of 5.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
5^i	1	5	2	10	4	20	8	17	16	11	9	22	18	21	13	19	3	15	6	7	12	14

Table 1: Powers of 5 in $\mathbf{Z}_{23}$.

So, we can read off the discrete logarithm to the base 5 of any element of $\mathbf{Z}_{23}^*$ from this table. For example, if we want to find the discrete logarithm of 17, we see that it appears under $i = 7$, i.e. $5^7 = 17$. So, $\log_5 17 = 7$. The table of logarithms is in Table 2.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
$\log_5 i$	0	2	16	4	1	18	19	6	10	3	9	20	14	21	17	8	7	12	15	5	13	11

Table 2: Logarithms to the base 5 in Z_{23}.

* * *

Recall that the natural logarithm, $\ln n$, and the logarithm to the base 10, $\log n$, are continuous functions of n. For example $\log 20 = 1.301029996$ and $\log 21 = 1.322219295$ is quite close to this. So, we can make use of continuity and differentiability of these functions to calculate logarithms and antilogarithms using numberical methods. However, as you can see from Table 2, this is not the case for discrete logarithms. In Table 2, while the discrete log of 2 to the base 5 is 2, the discrete log of 3 to the base 5 is 16, which is nowhere near 2. It increases and decreases with n in such a way that we can't use numerical methods to calculate discrete logarithms. You can see the comparison of discrete logarithm to the base 5 and the usual logarithm to the base five in Fig. 2. The red dots correspond to the values of the discrete logarithm to the base five and the values of $\log_5 n$ are joined by a blue line. As you can see, the values of the discrete logarithm are all over the place, but the values of $\log_5 n$ are continuous.

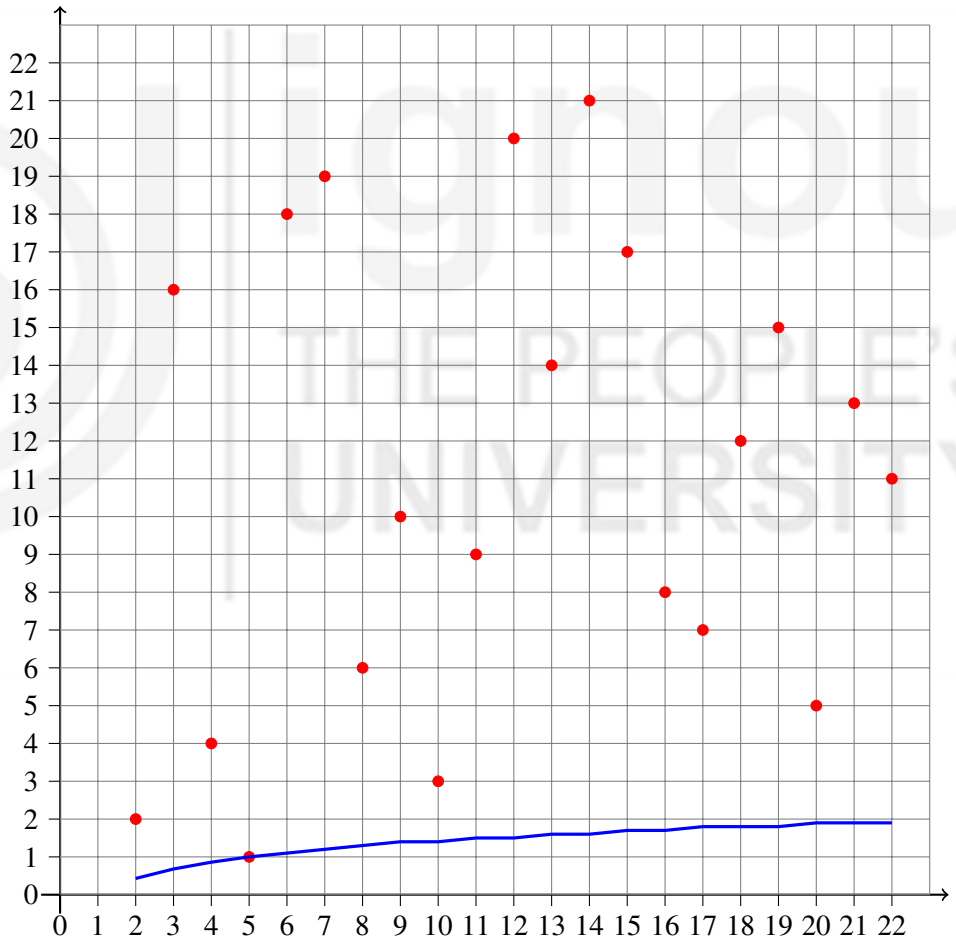

Fig. 2: Plot of Discrete logarithm (mod 23) to the base 5 and $\log_5 n$

We can always make a table like Table 2, but the amount of computation grows large very quickly with increase in p. If all the prime factors of $p-1$ are sufficiently small, there are other methods for calculating discrete logarithms. However, the calculation of discrete logarithms is very difficult if we choose our p so that $p-1$ has at least one large prime factor. We exploit this property for creating one-way functions in the following sections. In the next subsection, we will see how Alice and Bob can use this property of discrete logarithms to safely exchange keys for a symmetric key cryptosystem.

7.4.1 Diffie-Hellman key exchange

As we mentioned earlier, one problem in symmetric key cryptosystem is the problem of key exchange. We now describe a method by which Bob and Alice can decide upon a key without meeting in person.

1) Alice chooses a large prime p such that $p-1$ has a large prime factor. She then chooses a primitive root $\alpha \pmod p$. She selects an integer x, $0 < x < p-1$. She keeps the value of x secret, finds $\alpha^x \pmod p$ and sends the values (p, α, α^x) to Bob.

2) Bob chooses a y, $0 < y < p-1$ and keeps it secret. He finds $\alpha^y \pmod p$ and sends it to Alice.

3) Alice computes $\alpha^{xy} = (\alpha^y)^x$ from the value of α^y since she knows x. Bob computes $\alpha^{xy} \pmod p = (\alpha^x)^y \pmod p$ since he knows y.

Of course, Bob can also initiate the process by choosing the prime p, a primitive root $\alpha \pmod p$ and y, $0 < y < p-1$ and sending (p, α, α^y) to Alice. Either way, at the end of the exchanges, both Alice and Bob have the number α^{xy} and they can use this as the key for some symmetric key cryptosystem like AES.

Let us see why this system is secure. Eve has to calculate $\alpha^{xy} \pmod p$ from the value of α, α^x and α^y. This is a computationally infeasible problem known as the **Computational Diffie-Hellman problem.**

Of course, if Eve can compute discrete logs, she can find x from α^x and compute $(\alpha^y)^x = \alpha^{xy}$ from the given value of α^y and x. As we will see in the section on factorisation and discrete logarithms, if $p-1$ has a large prime factor, it is difficult to find discrete logarithim $\bmod p$. So, this problem is no harder than the problem of computation of discrete logs. However, it is not known whether solving Computational Diffie-Hellman problem is equivalent to finding Discrete logs.

Let us look at an example to understand the Diffie-Hellman key exchange.

Example 4: Suppose Alice chooses $p = 127$ and $\alpha = 3$. She chooses $x = 71$ and computes $3^{71} \pmod{127} \equiv 43 \pmod{127}$. She then sends the values $(127, 3, 43)$ to Bob. Bob chooses $y = 85$, computes $3^{85} \equiv 57 \pmod{127}$ and sends the value 57 to Alice. Alice finds $57^{71} \pmod{127}$ and gets 29. Bob finds $43^{85} \pmod{127}$ and gets the same number 29. So, Alice and Bob have the same number at the end of the process. Eve can't find out this number with the information available to her.

* * *

Here is an exercise to check your understanding of the above example.

E5) Suppose Alice chooses $p = 127$ and $\alpha = 3$ and computes $3^7 \equiv 28 \pmod{127}$. She starts the Diffie-Hellman key exchange process by sending $(127, 3, 28)$ to Bob. Bob responds by sending $3^9 \equiv 125 \pmod{127}$. At the end of the key exchange process, what number will both of them have?

In the next subsection, we will discuss the ElGamal Cryptosystem. This also uses discrete logarithm.

7.4.2 ElGamal Cryptosystem

In the case of ElGamal cryptosystem, we work in the field $\mathbf{F}_p^*$. This is our ciphertext space $\mathscr{C}$. To setup such a system, Bob does the following:

1) He chooses a large prime p such that $p-1$ has a large prime factor. We will see why it is necessary for $p-1$ to have a large prime factor later. He also chooses an arbitrary primitive root g mod p, i.e $g \in \mathbf{F}_p$ such that the cyclic group generated by g is the whole of $\mathbf{F}_p^*$.

2) He chooses at random an integer x such that $0 < x \leq p-2$, computes $y = g^x$, publishes (p, g, y) and keeps x a secret.

If Alice wants to send a message to Bob, she chooses at random a natural number k such that $1 \leq k \leq p-2$. She keeps the value of k secret and sends the pair $\left(g^k, \mathscr{M} y^k\right)$ to Bob. Since Bob knows x, he can compute $\left(g^k\right)^x = (g^x)^k = y^k$. Bob can easily find $y^{-k} = \left(y^k\right)^{-1}$ using extended euclidean algorithm. So, he can recover the message by multiplying $\mathscr{M} y^k$ by y^{-k}:

$$\mathscr{M} \equiv \mathscr{M} y^k y^{-k} \pmod{p}$$

Note that, strictly speaking, we cannot put this in the framework of Definition 2. This is because the same message will encrypt to different ciphertexts at different times depending on the value of k that Alice selects. However, apart from this, in essence, it is a public key cryptosystem in the sense that anyone can send a message to Bob using the publicly available values of p, g and y. Also, we can encrypt and decrypt fast using the repeated squaring algorithm.

If Eve wants to recover the message from $\left(g^k, \mathscr{M} y^k\right)$, she has to calculate $y^k = g^{xk}$ from g^x and g^k. As we saw earlier, this is the computational Diffie-Hellman problem and this is computationally hard to solve.

Let us see how we can find a primitive root $\bmod p$. Note that, we can check that g is a primitive root for a prime p as follows: Suppose $p-1 = \prod_{j=1}^{k} q_j^{n_j}$ is the factorisation of $p-1$ into distinct powers. Let $t_i = \prod_{j=1, j\neq i}^{k} q_j^{n_j}$ and $g_i = g^{t_i}$ for each i, $1 \leq i \leq k$. Then g is a primitive root $\bmod p$ iff $g_i^{q_i^{n_i-1}} \not\equiv 1 \pmod{p}$ for each value of i. This is a simple exercise in group theory. We leave this to you as an exercise to verify this fact.

We can put this in a slightly different way. We have

$$g_i = g^{t_i}$$
$$\therefore g_i^{q_i^{n_i-1}} = g^{t_i q_j^{n_i-1}}$$

But,

$$t_i q_i^{n_i-1} = \left(\prod_{j\neq i}^{k} q_j^{n_j}\right) q_i^{n_i-1} = \frac{p-1}{q_i}$$

Writing $s_i = \frac{p-1}{q_i}$, we have the following criterion:

Proposition 1: Let p be a prime and suppose

$$p-1 = \prod_{j=1}^{k} q_j^{n_j}$$

Then, $g \in \mathbf{F}_p^*$ is a primitive root if and only if $g^{s_i} \not\equiv 1 \pmod{p}$ for each i, $1 \leq i \leq k$ where $s_i = \frac{p-1}{q_i}$.

Let us now look at an example.

Example 5: Suppose Bob selects 109 as the prime p. Then, $p-1 = 108 = 2^2 \cdot 3^3$. Let $q_1 = 2$, $q_2 = 3$. Then $s_1 = 2 \cdot 3^3$ and $s_2 = 2^2 \cdot 3^2$.

Let us now check if 2 is a primitive root. We have $g^{s_1} = 2^{54} = 2^{54} \equiv 108 \equiv -1 \not\equiv 1 \pmod{109}$. But, $g^{s_2} = 2^{36} \equiv 1 \pmod{109}$. So, 2 is not a primitive root.

Let us check if 3 is a primitive root. We have $3^{s_1} = 3^{54} \equiv 1 \pmod{109}$. So, 3 is not a primitive root. We need not check 4 because 4 is a perfect power.

Let us check if 5 is a primitive root. We have $5^{s_1} = 5^{54} \equiv 1 \pmod{109}$, so 5 is not a primitive root.

Let us now try 6. We have

$$g^{s_1} = 6^{54} \equiv 108 \equiv -1 \pmod{109} \text{ and } g^{s_2} = 6^{36} \equiv 63 \not\equiv 1 \pmod{109}.$$

So, 6 is a primitive root $\bmod 109$.

Suppose Bob chooses $x = 40$. Then, he calculates $g^{40} = 6^{40} \equiv 7 \pmod{109}$. He keeps the value $x = 40$ secret and publishes the values $p = 109$, $g = 6$ and $g^{40} = 7$.

Suppose Alice wants to send the message $\mathscr{M} = 17$ to Bob. She chooses $k = 13$ and finds

$$g^k = 6^{13} \equiv 96 \pmod{109} \text{ and } g^{xk} = (g^x)^k = 7^{13} \equiv 80 \pmod{109}.$$

She then finds $\mathscr{M} g^{xk} = 17 \cdot 80 \equiv 52 \pmod{109}$. She then sends $(96, 52)$ to Bob. To recover the message Bob computes $\left(g^k\right)^{p-1-x} = 96^{109-1-40} = 96^{68} \equiv 15 \pmod{109}$. He then multiplies the encrypted message 52 by 15 to get $52 \cdot 15 = 780 \equiv 17 \pmod{109}$, thus recovering the message.

* * *

E6) Let G be a finite abelian group and let n divide the order of G. Suppose $n = \prod_{j=1}^{k} q_j^{n_j}$ and let $g \in G$. Write

$$t_i = \prod_{j=1, j\neq i}^{k} q_j^{n_j} \text{ and } g_i = g^{t_i}.$$

Then, g has order n if and only if $g^n = 1$ and g_i has order $q_i^{n_i}$ for each i, i.e.

$$g_i^{q_i^{n_i-1}} \neq 1 \text{ and } g_i^{q_i^{n_i}} = 1 \text{ for all values of j.}$$

E7) Suppose Bob chooses $p = 181$. Check that 2 is a primitive root $\bmod 181$. Suppose Bob chooses $x = 21$ and publishes $(181, 2, 86)$. He receives the pair $(7, 122)$. Find the message.

Recall that the security of RSA cryptosystem depends on the fact that it is difficult, in general, to factorise a natural number. Similarly, the security of the ElGamal cryptosystem depends on the fact that it is difficult find disrete logs in $\mathbf{Z}_p^*$, p a prime. In the next section, we will see that we can solve these problems under certain conditions using certain methods. We will gain some insights by studying the conditions under which these problems can be solved and the methods by which they can be solved. This insight is of help to us in strengthening these cryptosystems.

7.5 INTEGER FACTORISATION AND DISCRETE LOGARITHMS

In this section, we will wear the hat of a cryptanalyst and look at methods for breaking the RSA and the ElGamal cryptosystem. We will discuss the problems of integer factorisation and discrete logarithms, the problems a cryptanalyst has to solve if she has to break the RSA and ElGamal cryptosystems, respectively. Our discussion follows [15].

We start with integer factorisation. For all practical purposes, the method of dividing an integer n by all primes $p \leq \sqrt{n}$ is much too slow. In this section, we will discuss other methods which are faster. Before we move on to more general methods, we would like to amplify our remark that we can factorise the modulus n used in RSA cryptosystem if we know the decryption exponent.

We mentioned while discussing the RSA cryptosystem that we can factor n if we know both the encryption and decryption exponent. In [8], Miller proved that we can successfully factorise n, with a high probability, if we know any multiple of $\phi(n)$. Since $ed \equiv 1 \pmod{\phi(n)}$, $ed-1$ is a multiple of $\phi(n)$. So, we can apply Miller's method to factorise n if we know e and d. We will discuss this method now.

Here is how the probabilistic algorithm works. Suppose we can find an exponent r such that

$$b^r \equiv 1 \pmod{n} \text{ for all b with } (b,n) = 1. \tag{11}$$

Note that, any multiple of $\phi(n)$ will satisfy this condition since $a^{\phi(n)} \equiv 1 \pmod{n}$ for all a with $(a,n) = 1$, by Euler's theorem. If e and d are the encryption and decryption exponents, we have $b^{ed-1} \equiv 1 \pmod{n}$ for all b with $(b,n) = 1$. So, there is always an r satisfying Eqn. (11).

Write $r = a2^s$ with a odd. Choose a random b with $1 < b < n-1$. If $\gcd(b,n) \neq 1$ we have a factor of n. Otherwise, let $b_0 = b^a \pmod{n}$. We find $b_1, b_2, \ldots$, in **N** such that

$$b_1 \equiv b_0^2 \pmod{n},\ b_2 \equiv b_1^2 \pmod{n},\ b_3^2 \equiv b_2 \pmod{n},$$

etc. If $b_0 \equiv 1 \pmod{n}$, we choose another b and repeat the procedure. Also, if $b_k \equiv -1 \pmod{n}$ for some k, we choose a different b and repeat the procedure. If $b_{k+1} \equiv 1 \pmod{n}$ and $b_k \not\equiv \pm 1 \pmod{n}$ for some k, $(b_k - 1, n)$ gives a non-trivial divisor of n. This is because, $b_k^2 \equiv b_{k+1} \equiv 1 \pmod{n}$ or $(b_k-1)(b_k+1) = b_k^2 - 1 \equiv 0 \pmod{n}$. So, $n \mid (b_k+1)(b_k-1)$. Since $b_k \not\equiv \pm 1 \pmod{n}$, $n \nmid b_k - 1$ and $n \nmid b_k + 1$. So, $(n, b_k - 1)$ and $(b_k + 1, n)$ must be non-trivial factors of n. Also, since $b_s \equiv 1 \pmod{n}$, for each b with $(b,n) = 1$, it is guaranteed that there is a k such that $b_k \equiv 1 \pmod{n}$.(Why?)

Although it is only a probabilistic algorithm, the probability of finding a non trivial factor using this algorithm is high. So, if the decryption exponent leaks out, changing only e and d is not enough. We have to start from scratch with a new set of values for p and q and choose new values for e and d corresponding to the new value of n.

Let us now look at an example.

Example 6: Suppose $n = 667$, $e = 39$, $d = 79$. We have $39 \cdot 79 - 1 = 2^3 \cdot 385$. Let us take $b = 3$. Then, $(3, 667) = 1$. We have

$$b_0 = 3^{385} \equiv 162 \pmod{667}$$
$$b_1 \equiv b_0^2 \equiv 231 \pmod{667}$$
$$b_2 \equiv b_1^2 \equiv 1 \pmod{667}$$

We have $b_2 \equiv 1 \pmod{667}$ and $b_1 \not\equiv \pm 1 \pmod{667}$. Also, $(b_1 - 1, 667) = (230, 667) = 23$. So, $667 = 23 \cdot 29$.

* * *

Here is an exercise for you to try.

E8) Let $n = 12193$, $e = 303$, $d = 79$. You can take for granted the following information:

$$b_0 \equiv 2^{187} \equiv 9079 \pmod{12193}$$
$$b_1 \equiv 9079^2 \equiv 3561 \pmod{12193}$$
$$b_2 \equiv 3561^2 \equiv 1 \pmod{12193}$$

Factor n.

Remark 2: Even if we find one b and one $r > 0$, where b and r are integers with $b^r \equiv 1 \pmod{n}$, we may succeed in factoring n. We write $r = 2^s a$ with a odd. We then find $b_0 = b^a$, $b_1 \equiv b_0^2 \pmod{n}$, $\cdots$ and proceed exactly as before. Of course, we may get $b_k \equiv -1 \pmod{n}$, for some $k \leq s - 1$ and this method may fail.

Another method we mentioned in our discussion of the precautions we have to take while implementing the RSA algorithm is Pollard's $p - 1$ algorithm. Pollard invented this algorithm in 1974. Using this method, we can factorise a number n if there is a prime p which divides n and $p - 1$ has only small prime factors. Here is a brief description of the method.

The $p - 1$ Factoring Algorithm. Choose an integer $a > 1$. We often choose $a = 2$. We choose a bound B and compute $b \equiv a^{B!} \pmod{n}$ as follows. Let $b_1 \equiv a \pmod{n}$ and $b_j \equiv b_{j-1}^j \pmod{n}$. Then $b_B \equiv b \pmod{n}$. Let $d = \gcd(b - 1, n)$. If $1 < d < n$, we have found a nontrivial factor of n.

Suppose p is a prime factor of n such that $p - 1$ has only small prime factors. Then $B!$ is likely to be divisible by $p - 1$, say $B! = (p - 1)k$. We have

$$b \equiv a^{B!} \equiv (a^{p-1})^k \equiv 1 \pmod{p}$$

by Fermat's theorem, so p will occur in the greatest common divisor of $b - 1$ and n. If q is another prime factor of n, it is unlikely that $b \equiv 1 \pmod{q}$, unless $q - 1$ also has only small prime factors. If $d = n$, not all is lost. In this case, we have an exponent r (namely $B!$) and an a such that $a^r \equiv 1 \pmod{n}$. There is a good chance that the method due to Miller that we mentioned in the previous will yield a factor of n.

Let us look at an example.

Example 7: Consider the number $n = 162983$. Let us choose $B = 30$ as the bound. We have $2^{30!} \equiv 155655 \pmod{n}$ and $(155654, n) = 349$, yielding a factor. By division, we get the other factor 467.

* * *

Try the next exercise to check your understanding of the above example.

E9) Let $n = 16867$. Factor n using $p - 1$-algorithm, taking the value of B to be 10.

In the previous section, we mentioned a method which is attributed to Fermat called **Fermat factorisation method**. Dividing by powers of two if necessary, we can always

assume that n is odd. In this method, we try find positive integers x and y such that $x^2 - y^2 = n$. We then have $(x+y)(x-y) = n$. If $x - y > 1$, we have a factor of n, namely $x - y$. Conversely, suppose $n = pq$ where p and q are odd numbers, $p, q > 1$. We have $n = (x+y)(x-y)$ if we take $x = \frac{p+q}{2}$, $y = \frac{p-q}{2}$. Thus, if n is odd, it is composite if and only if $n = (x+y)(x-y)$ for $x, y > 1$, $x, y \in \mathbf{N}$.

Given an odd integer n, how do we find x and y? For various values of i, we check whether $n + i^2$ is a perfect square. As we saw just now, if there is a factorisation $n = pq$, then $n = x^2 - y^2$ with $x = \frac{p+q}{2}$, $y = \frac{p-q}{2}$. So, the number of values of i we have to check will be at most

$$\min\left\{ \frac{p-q}{2} \,\middle|\, n = pq, \text{ where } p, q \in \mathbf{N}, p, q > 1 \right\}$$

If there are factors p and q that are close to each other, $p - q$ is small and so n can be factored easily. Let us now look at an example.

Example 8: Suppose $n = 66013$. We have $n + 1 = 66014$, $n + 2^2 = 66017$, …, $n + 5^2 = 66038$ which are not perfect squares, but $n + 6^2 = 66049 = 257^2$. Therefore, $n = 257^2 - 6^2 = (257+6)(257-6) = 251 \cdot 263$.

* * *

Try the following exercise to check your understanding.

E10) Factor 51067 using Fermat's method.

Kraitchik observed that it is not necessary find x and y such that $x^2 - y^2 = n$; it is enough to find x and y such that $x^2 \equiv y^2 \pmod{n}$ with $x \not\equiv y \pmod{n}$. If $1 < (x-y, n) < n$, then $d = (x-y, n)$ is a proper divisor of n. Pomerance and others developed the quadratic sieve algorithm using this idea. (See [11] for an account of this algorithm.) The number field sieve, a generalisation of the quadratic sieve has already superseded the quadratic sieve. Still, quadratic sieve is of interest because the ideas behind this are used in the number field sieve. We will now discuss a simple algorithm that is quite similar to the quadratic sieve algorithm. Our algorithm will be useful in understanding the main ideas behind the quadratic sieve algorithm. We will follow [15] in the treatment of this algorithm.

We present the algorithm through an example.

Example 9: Suppose we want to factor $n = 7680587$. We fix a limit B. The set of primes less than B is called our **factor base**. We fix $B = 20$ for our example. So, our factor base consists of the primes 2, 3, 5, 7, 11, 13, 17, 19. We find integers x_1, x_2, …, x_ℓ such that the prime divisors of $x_i^2 \pmod{7680587}$ are less than B; we will see how to find such numbers later. For example, $3929^2 \equiv 75867 \pmod{7680587}$ and $75867 = 3 \cdot 11^3 \cdot 19$, so 3929 is one such number. If there are q primes $\{p_1, p_2, \ldots, p_q\}$ in the factor base, we form a $\ell \times q$ matrix as follows: Each column of the matrix corresponds to a prime in the factor base and each row corresponds to a x_i such that $x_i^2 \equiv c_i \pmod{7680587}$ with all the prime divisors of c_i less than or equal to B. The entry in the i^{th} row, j^{th} column gives the power of the j^{th} prime that divides c_i. For example, since $3929^2 \equiv 75867 \pmod{7680587}$ and $75867 = 3 \cdot 11^3 \cdot 19$, 3929 will correspond to the row $[0,1,0,0,3,0,0,1]$ indicating that the power of 3 dividing 75867 is 1, the power of 11 dividing 75867 is 3 and the power of 19 dividing 75867 is 1 and

the primes 2, 5, 7, 13 and 17 do not divide 75867. Similarly, we find that

$$\begin{aligned}
4801^2 &\equiv 7840 \pmod{7680587} & 7840 &= 2^5 \cdot 5 \cdot 7^2 \\
7839^2 &\equiv 5225 \pmod{7680587} & 5225 &= 5^2 \cdot 11 \cdot 19 \\
9602^2 &\equiv 31360 \pmod{7680587} & 31360 &= 2^7 \cdot 5 \cdot 7^2 \\
9997^2 &\equiv 92378 \pmod{7680587} & 92378 &= 2 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \\
14403^2 &\equiv 70560 \pmod{7680587} & 70560 &= 2^5 \cdot 3^2 \cdot 5 \cdot 7^2 \\
14932^2 &\equiv 227601 \pmod{7680587} & 227601 &= 3^2 \cdot 11^3 \cdot 19
\end{aligned}$$

We obtain the matrix

$$\begin{array}{c|cccccccc}
 & 2 & 3 & 5 & 7 & 11 & 13 & 17 & 19 \\
\hline
3929 & 0 & 1 & 0 & 0 & 3 & 0 & 0 & 1 \\
4801 & 5 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\
7839 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 1 \\
9602 & 7 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\
9997 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
14403 & 5 & 2 & 1 & 2 & 0 & 0 & 0 & 0 \\
14932 & 0 & 2 & 0 & 0 & 3 & 0 & 0 & 1
\end{array}. \tag{12}$$

To find numbers like 3929, 4801, …, we look at the numbers $\left[\sqrt{\mathrm{in}}+\mathrm{j}\right]$ for small values of j. Here [x] denotes the greatest integer less than or equal to x. The square of such a number is approximately $\mathrm{in}+2\mathrm{j}\sqrt{\mathrm{in}}+\mathrm{j}^2$, which is approximately $2\mathrm{j}\sqrt{\mathrm{in}}+\mathrm{j}^2$ mod n. As long as i is not too large, this number is fairly small, hence there is a good chance it is a product of small primes. For example, if we take $\mathrm{i}=3$, $\mathrm{j}=1$, we have

$$\sqrt{3 \cdot 7680587}+1 = 4801.1834 \text{ and } [4801.1834] = 4801.$$

We have

$$4801^2 \equiv 7840 \text{ and } 7840 = 2^5 \cdot 5 \cdot 7^2$$

so, we get the second row of the matrix in Eqn. (12).

We look for linear dependencies mod 2 among the rows of Eqn. (12). We reduce the rows of the matrix in Eqn. (12) mod 2 to get matrix

$$\begin{bmatrix}
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}. \tag{13}$$

We note that second, fourth and sixth rows are identical. Also, third and the seventh rows are identical. Since we are working modulo 2, any two of them add up to $\mathbf{0}$. A systematic way of finding relations is to view Eqn. (13) as map from $\mathbf{F}_2^7$ to $\mathbf{F}_2^8$ and find the relations from the kernel. For example, the kernel of the matrix in Eqn. (13) is generated by the vectors $\{(0,1,0,1,0,0,0),(0,1,0,0,0,1,0),(0,0,1,0,0,1)\}$. From the kernel, we have the following relations:

1. Note that, on multiplying Eqn. (13) on the left by the first vector $(0,1,0,1,0,0,0)$, we get the sum of the second and fourth columns as the answer. Since $(0,1,0,1,0,0,0)$ is in the kernel, this implies that the sum of the second and fourth rows are zero.

2. Similarly, from the second vector in the kernel we get that the sum of the second and sixth rows are zero.
3. From the third vector in the kernel we get that the sum of the third and eighth rows are zero.

When we have such a dependency, we get a relation of the form

$$x^2 \equiv y^2 \pmod{7680587}.$$

For example:

1. The sum of the second and fourth rows of the matrix in Eqn. (13) is **0**. The second row corresponds to the relation

$$4801^2 \equiv 2^5 \cdot 5 \cdot 7^2 \pmod{7680587}$$

and the fourth row corresponds to the relation

$$9602^2 \equiv 2^7 \cdot 5 \cdot 7^2 \pmod{7680587}.$$

Recall that, if
$a \equiv b \pmod{n}$
and
$c \equiv d \pmod{n}$, then
$ac \equiv bd \pmod{n}$.

Multiplying the congruences

$$4801^2 \equiv 2^5 \cdot 5 \cdot 7^2 \pmod{7680587} \text{ and } 9602^2 \equiv 2^7 \cdot 5 \cdot 7^2 \pmod{7680587}$$

we get the congruence

$$(4801 \cdot 9602)^2 \equiv 2^{12} \cdot 5^2 \cdot 7^4 \equiv \left(2^6 \cdot 5 \cdot 7^2\right)^2 \pmod{7680587}$$

2. The sum of the second and sixth rows of the matrix in Eqn. (13) is **0**. The second row corresponds the congruence

$$4801^2 \equiv 2^5 \cdot 5 \cdot 7^2 \pmod{7680587}$$

and the sixth row corresponds to the congruence

$$14403^2 \equiv 2^5 \cdot 3^2 \cdot 5 \cdot 7^2 \pmod{7680587}.$$

Multiplying the congruences

$$4801^2 \equiv 2^5 \cdot 5 \cdot 7^2 \pmod{7680587} \text{ and } 14403^2 \equiv 2^5 \cdot 3^2 \cdot 5 \cdot 7^2 \pmod{7680587}$$

we get the congruence

$$(4801 \cdot 14403)^2 \equiv \left(2^5 \cdot 3 \cdot 5 \cdot 7\right)^2 \pmod{7680587}.$$

3. The sum of the third and sixth rows of the matrix in Eqn. (13) is **0**. The third row corresponds to the congruence

$$7839^2 \equiv 5^2 \cdot 11 \cdot 19 \pmod{7689587}$$

and the seventh row corresponds to the congruence

$$14932^2 \equiv 3^2 \cdot 11^3 \cdot 19 \pmod{7680587}$$

Multiplying the congruences

$$7839^2 \equiv 5^2 \cdot 11 \cdot 19 \pmod{7680587} \text{ and } 14932^2 \equiv 3^2 \cdot 11^3 \cdot 19 \pmod{7680587}$$

we get the congruence

$$(7839 \cdot 14932)^2 \equiv \left(3 \cdot 5 \cdot 11^2 \cdot 19\right)^2 \pmod{7680587}$$

Therefore, we have found three relations of the form $x^2 \equiv y^2 \pmod{n}$. If $x \not\equiv \pm y \pmod{n}$, then $\gcd(x-y, n)$ yields a nontrivial factor of n. If $x \equiv \pm y \pmod{n}$, then $\gcd(x-y, n) = 1$ or n, so we don't obtain a factorisation.

1. We have

$$(4801 \cdot 9602)^2 \equiv \left(2^6 \cdot 5 \cdot 7^2\right)^2 \pmod{7680587},$$
$$\text{but } 4801 \cdot 9602 \equiv 2^6 \cdot 5 \cdot 7^2 \pmod{7680587}.$$

   So, this relation does not yield a factor of 7680587.

2. We have

$$(4801 \cdot 14403)^2 \equiv \left(2^5 \cdot 3 \cdot 5 \cdot 7^2\right)^2 \pmod{7680587}$$
$$\text{but } (4801 \cdot 14403) \equiv 2^5 \cdot 3 \cdot 5 \cdot 7^2 \pmod{7680587}.$$

   So, this relation does not yield a factor 7680587.

3. We have

$$(7839 \cdot 14932)^2 \equiv \left(3 \cdot 5 \cdot 11^2 \cdot 19\right)^2 \pmod{7680587}$$
$$\text{and } (7839 \cdot 14932) \not\equiv 3 \cdot 5 \cdot 11^2 \cdot 19 \pmod{7680587}.$$

   We have $(7839 \cdot 14932 - 3 \cdot 5 \cdot 11^2 \cdot 19, 7680587) = 1129$. Thus, we have found a factor of 7680587. Dividing 7680587 by 1129, we get the other factor 6803. You can check that 1129 and 6803 are primes.

Note that, for the sake of simplicity, we have taken only seven rows. But, in practice, we usually make sure there are more rows than columns, i.e. $\ell > q$, so that the kernel of the reduced matrix is always non-trivial.

* * *

Try the following exercise to check your understanding of the above example.

E11) Factorise $n = 115697$ using the following relations:

$$486^2 \equiv 2 \cdot 7^4 \pmod{115697}$$
$$972^2 \equiv 2^3 \cdot 7^4 \pmod{115697}$$
$$1407^2 \equiv 2^9 \cdot 5^2 \pmod{115697}$$

The quadratic sieve is an improved version of the above method. The number field sieve uses more sophisticated techniques to produce relations of the form $x^2 \equiv y^2 \pmod{n}$ and is somewhat faster in many situations. See [7] for an exposition of this method.

Next, we shall discuss methods for finding discrete logarithms. You may like to review the relevant portions from Block 1 and Sec. 4 of this Unit before proceeding further.

7.5.1 Computing Discrete Logarithms

Let us fix a prime p and suppose α and β be non-zero integers. Suppose that we have to find x such that

$$\alpha^x \equiv \beta \pmod{p}. \tag{14}$$

Recall that, the problem of finding x is called the **discrete logarithm** problem because if x satisfies Eqn. (14), then we say that x is the discrete logarithm of β to the base α. Of course, a solution may not exist for arbitrary α and β. However, if we take α to be a

primitive root mod p, there is always a solution to Eqn. (14) for any nonzero β. So, from now, we will assume that α is a primitive root mod p for the sake of simplicity. Observe that,

$$\alpha^{m_1} \equiv \alpha^{m_2} \pmod{p} \Leftrightarrow m_1 \equiv m_2 \pmod{p-1}. \tag{15}$$

Note that, we can formulate the problem in Eqn. (14) in a more general setting. Suppose G is a cyclic group of order n and α is a generator of G. If $\beta \in G$, we can ask for the solution to the equation

$$\alpha^x = \beta. \tag{16}$$

The equation has a solution that lies between 0 and $n-1$.

We now describe a method due to Pohlig and Hellman for solving Eqn. (16). See [10]. Pohlig and Hellman discuss a method for solving Eqn. (14) on the previous page, but we will discuss the method in the more general situation given by Eqn. (16). Our will work for Eqn. (14) because $\mathbf{Z}_p^*$ is a cyclic group of order $p-1$.

We first show how to reduce the problem in Eqn. (16) to the case where the cyclic group has prime power order. Suppose

$$n = \prod_i q_i^{r_i}$$

is the factorisation of n into primes. Set

$$n_i = \frac{n}{q_i^{r_i}}, \alpha_i = \alpha^{n_i} \text{ and } \beta_i = \beta^{n_i}.$$

Then, the order of α_i is $q_i^{r_i}$ and it generates the unique Sylow-q_i subgroup of G.(Why?) Further, order of β_i divides $q_i^{r_i}$ and so β_i is in the Sylow-q_i subgroup of G. So, the equation

$$\alpha_i^x = \beta_i \tag{17}$$

has a solution for each i, $1 \leq i \leq k$. Let $x(i)$, $1 \leq i \leq k$, be a solution of Eqn. (17). The next proposition says how to get the solution to Eqn. (16) from the solutions to the family of equations given by Eqn. (17).

Proposition 2: Suppose $x(i)$ is a solution to Eqn. (17) for $1 \leq i \leq k$ and $x \in \{0, 1, \ldots, n-1\}$ is a solution to the set of simultaneous congruences $x \equiv x(i) \pmod{q_i^{r_i}}$. Then, x is a solution to Eqn. (16).

Proof: We have

$$\begin{aligned}
\left(\alpha^{-x}\beta\right)^{n_i} &= (\alpha^{n_i})^{-x}\beta^{n_i} \equiv \alpha_i^{-x}\beta_i \\
&= \alpha_i^{-x(i)+kq_i^{r_i}}\beta_i, k \in \mathbf{Z} \quad (\because x \equiv x(i) \pmod{q_i^{r_i}}) \\
&= \alpha_i^{-x(i)}\beta_i \quad (\because \alpha_i \text{ has order } q_i^{r_i}) \\
&= 1
\end{aligned}$$

where we denote the identity element of G by 1. Thus, order of $(\alpha^{-x}\beta)$ divides n_i for all i. So, the order divides the gcd of n_i. But, the gcd of n_i is one, so the order of $\alpha^{-x}\beta$ is one, i.e. $\alpha^{-x}\beta = 1$ or $\alpha^x \equiv \beta$. ■

Next, we show how to compute $x(i)$ for $1 \leq i \leq k$.

We write $x(i)$ as

$$x(i) = x_0(q_i) + x_1(q_i)q_i + x_2(q_i)q_i^2 + \cdots + x_{r_i-1}(q_i)q_i^{r_i-1}. \tag{18}$$

where

$$0 \leq x_j(q_i) \leq q_i - 1 \text{ for } 0 \leq j \leq r_i - 1.$$

We'll determine the coefficients $x_0(q_i), x_1(q_i), \ldots x_{r_i-1}(q_i)$ successively, and thus obtain x mod $q_i^{r_i}$.

For simplifying notation, we will fix a q_i and write q for q_i and k for r_i. We will also write x_0, x_1, … instead of $x_0(q_i)$, $x_1(q_i)$, ….

Raising both sides of Eqn. (17) to the power q^{k-1}, we get

$$\alpha_i^{xq^{k-1}} = \beta_i^{q^{k-1}} \tag{19}$$

From Eqn. (18), we get

$$q^{k-1}x = x_0q^{k-1} + q^k\left(x_1 + x_2q + \cdots + x_{k-1}q^{k-2}\right) = x_0q^{k-1} + q^km \tag{20}$$

where m is an integer. Since α_i has order q^k, it follows that

$$\alpha_i^{xq^{k-1}} = \alpha_i^{x_0q^{k-1}}\alpha_i^{mq^k} = \alpha_i^{x_0q^{k-1}}. \tag{21}$$

Writing $\gamma_i = \alpha_i^{q^{k-1}}$ and $\delta_0(q) = \beta_i$, we can write Eqn. (21) as

$$\gamma_i^{x_0} = \delta_0(q)^{q^{k-1}} \tag{22}$$

where γ_i and $\delta_0(q)^{q^{k-1}}$ have order q and we have to solve Eqn. (22) in a group of order q. We solve this equation by brute force; we find different powers of γ_i and see which power of γ_i gives $\delta_0(q)^{q^{k-1}}$. The solution to Eqn. (22) will give us the value of x_0.

Suppose we have found x_0, $x_1, \ldots$, x_{i-1} and we want to find x_i, $1 \leq i \leq k-1$. We have

$$\alpha_i^{x_iq^i + \cdots + x_{k-1}q^{k-1}} = \beta\alpha_i^{-\left(x_0 + x_1q + \cdots + x_{i-1}q^{i-1}\right)} \tag{23}$$

Denoting the right hand side of Eqn. (23) by $\delta_i(q)$ and raising both sides of Eqn. (23) to the power q^{k-i-1}, we get

$$(\alpha_i)^{x_iq^{k-1} + x_{i+1}q^k + \cdots + x_{k-1}q^{2k-i-2}} = \delta_i(q)^{q^{k-i-1}} \tag{24}$$

The left hand side of Eqn. (24) is

$$\begin{aligned}\alpha_i^{x_iq^{k-1} + x_{i+1}q^k + \cdots + x_{k-1}q^{2k-i-2}} &= \alpha_i^{x_iq^{k-1} + m'q^k} \text{ for some integer } m'.\\ &= \alpha_i^{x_iq^{k-1}}\alpha_i^{m'q^k} = \alpha_i^{x_iq^{k-1}}\end{aligned}$$

since α_i has order q^k. So, denoting $(\alpha_i)^{q^{k-1}}$ by γ_i we can write Eqn. (24) as

$$(\gamma_i)^{x_i} = \delta_i(q)^{q^{k-i-1}} \tag{25}$$

and this is a discret logarithm problem with solution x_i. Since γ_i has order q, we have to solve Eqn. (25) in a group of order q. As we did in the case of x_0, we again solve for x_i by brute force.

Let us summarise the procedure for solving Eqn. (14).

1) Write $n = q_1^{r_1}q_2^{r_2}\cdots q_k^{r_k}$ and set $n_i = n/q_i^{r_i}$, $\alpha_i = \alpha^{n_i}$ and $\beta_i = \beta^{n_i}$.

2) Write $x(i) = x_0(q_i) + x_1(q_i)q_i + \cdots + x_{r_i-1}(q_i)q_i^{r_i-1}$. Solve for $x_0(q_i)$, $x_1(q_i)$, …, $x_{r_i-1}(q_i)$ as follows:

Write

$$\gamma_i = \alpha_i^{q_i^{r_i-1}}$$

To find $x_0(q_i)$ solve the equation

$$\gamma_i^x = \beta_i^{q_i^{r_i-1}} = \delta_0(q_i)^{q_i^{r_i-1}} \tag{26}$$

where we write $\delta_0(q_i)$ for β_i. Eqn. (26) is an equation in a subgroup of order q_i and we have to solve this to get $x_0(i)$.

After getting $x_0(q_i)$, $x_1(q_i)$, … $x_{\ell-1}(q_i)$, compute

$$\begin{aligned}\delta_\ell(q_i) &= \beta_i\alpha_i^{-\left(x_0(q_i)+x_1(q_i)q_i+\cdots+x_{\ell-1}(q_i)q_i^{\ell-1}\right)} \\ &= \delta_{\ell-1}(q_i)\alpha_i^{-x_{\ell-1}(q_i)q_i^{\ell-1}}, 1 \le \ell \le r_i - 1\end{aligned}$$

,

$$\boxed{\text{i.e. } \delta_\ell(q_i) = \delta_{\ell-1}(q_i)\alpha_i^{-x_{\ell-1}(q_i)q_i^{\ell-1}}, 1 \le \ell \le r_i - 1} \tag{27}$$

and solve the equation

$$\gamma_i^x = \delta_\ell(q_i)^{q_i^{r_i-\ell-1}}. \tag{28}$$

This will give the value of $x_\ell(q_i)$. Successively solving for $x_0(q_i)$, $x_1(q_i)$, …, $x_{r_i-1}(q_i)$, we will get $x(i) = x_0(q_i) + x_1(q_i)q_i + \cdots + x_{r_i-1}(q_i)q_i^{r_i-1}$.

3) After finding $x(1)$, $x(2)$, …, $x(k)$, solve the simultaneous congruences

$$x \equiv x(i) \pmod{q_i^{r_i}}$$

using Chinese Remainder Theorem to find the value of x.

Let us look at an example now.

Example 10: Suppose $p = 2269$. Here, the order of the group $\mathbf{Z}_p^*$ is 2268, i.e. $n = 2268$. Then, 2 is a primitive root. Suppose we want to solve the equation

$$2^x \equiv 6 \pmod{2269}.$$

We have $2268 = 2^2 \cdot 3^4 \cdot 7$. So,

$$\boxed{q_1 = 2, r_1 = 2, q_2 = 3, r_2 = 4, q_3 = 7, r_3 = 1.}$$

We also have

$$\boxed{n_1 = \tfrac{2268}{2^2} = 567, n_2 = \tfrac{2268}{3^4} = 28, n_3 = \tfrac{2268}{7} = 324.}$$

We have

$$\alpha_1 = \alpha^{n_1} = \alpha^{567} \equiv 1287 \pmod{2269} \text{ and } \beta_1 = 6^{567} = 982 \pmod{2269}. \tag{29}$$

$$\gamma_1 = \alpha_1^{q_1^{r_1-1}} = \alpha_1^2 = \alpha_1^2 = 1287^2 \equiv 2268 \pmod{2269}$$

Let $x(2) = x_0(2) + x_1(2) \cdot 2$. We have to solve the equation $\gamma_1^x = \beta_1^2$ to find the value of $x_0(2)$, i.e.

$$\gamma_1^x = \beta_1^2 \text{ or } 2268^x \equiv 2268 \pmod{2269}$$

to get $x_0(2)$. So, $x_0(2) = 1$. We have

$$\delta_1(2) = \beta_1 \alpha_1^{-x_0(2)} = \beta_1 \alpha_1^{-1} \equiv 982 \cdot 1287^{-1} \equiv 2268 \pmod{2269}$$

We have to solve the equation Eqn. (28) with $\ell = 1$ to get $x_1(2)$. In other words, we have to solve the equation

$$\gamma_1^x \equiv \delta_1^{2^{2-1-1}} = \delta_1 \text{ or } 2268^x \equiv 2268 \pmod{2269}$$

So, $x_1(2) = 1$. Therefore, $x(2) = 3$.

Let us now compute $x(3)$. We write

$$x(3) = x_0(3) + x_1(3) \cdot 3 + x_2(3) \cdot 3^2 + x_3(3) \cdot 3^3.$$

We have

$$\alpha_2 = \alpha^{n_2} = \alpha^{28},$$

so $\alpha_2 \equiv 2^{28} \equiv 1411 \pmod{2269}$. Further, $\gamma_2 = \alpha_2^{3^3}$ so

$$\gamma_2 \equiv 1411^{27} \equiv 2186 \pmod{2269}.$$

We have $\beta_2 = \beta^{n_2}$, so $\beta_2 \equiv 6^{28} \equiv 2252 \pmod{2269}$. To find $x_0(3)$, we have to solve Eqn. (26) with $q_2 = 3$, $r_2 = 4$, i.e.

$$2186^x \equiv 2252^{27} \pmod{2269} \text{ or } 2186^x \equiv 2186 \pmod{2269}$$

So $x_0(3) = 1$.

To find $x_1(3)$, we have to solve Eqn. (28) with $\ell = 1$. First, let us compute $\delta_1(3)$. We have

$$\delta_1(3) = \beta_2 \alpha_2^{-1} \text{ or } \delta_1 = 2252 \cdot 1411^{-1} \equiv 82 \pmod{2269}$$

We have

$$\delta_1(3)^{3^{4-1-1}} = \delta_1^9 \equiv 82^9 \equiv 1 \pmod{2269}$$

To find $x_1(3)$, we have to solve the equation

$$\gamma_2^x \equiv 1 \pmod{2269}$$

So, $x_1(3) = 0$. So,

$$\delta_2(3) = \delta_1(3) \equiv 82 \pmod{2269} \text{ and } \delta_2(3)^{3^{4-2-1}} = \delta_2(3)^3 \equiv 82^3 \equiv 1 \pmod{2269}$$

To find $x_2(3)$, we have to solve the equation

$$\gamma_2^x = \delta_2(3)^{3^{4-2-1}} \text{ or } \gamma_2^x = 1$$

So, $x_2(3) = 0$. Again,

$$\delta_3(3) = \delta_2(3) \equiv 82 \pmod{2269} \text{ and } \delta_3(3)^{3^{4-3-1}} = \delta_3(3) \equiv 82 \pmod{2269}.$$

To find $x_3(3)$, we have to solve the equation

$$\gamma_2^x = \delta_3(3) \text{ or } 2186^x \equiv 82 \pmod{2269}.$$

We have $2186^2 \equiv 82 \pmod{2269}$, so $x_3(3) = 2$. So,

$$x(3) = x_0(3) + x_1(3)\cdot 3 + x_2(3)\cdot 3^2 + x_3(3)\cdot 3^3 = 1 + 0\cdot 3 + 0\cdot 3^2 + 2\cdot 3^3 = 55.$$

Let us now compute $x(7)$. We have,

$$\alpha_3 = \alpha^{n_3} = 2^{324} \equiv 738 \pmod{2269} \text{ and } \beta_3 = \beta^{n_3} \equiv 6^{324} \equiv 84 \pmod{2269}$$

Further, here $r_3 = 1$. So, $x(7) = x_0(7)$ and $\gamma_3 = \alpha_3$. We have

$$\delta_0(7) = \beta_3^{7^{1-1}} = \beta_3.$$

So, we have to solve the equation

$$738^x \equiv 84 \pmod{2269}$$

We have $738^2 \equiv 84 \pmod{2269}$. So, $x(7) = 2$.

To find the value of x, we have to solve the congruences

$$x \equiv 3 \pmod 4$$
$$x \equiv 55 \pmod{81}$$
$$x \equiv 2 \pmod 7$$

using chinese remainder theorem. Solving, we get $x = 1675$.

* * *

Here is an exercise for you to try:

E12) Solve the equation $5^x \equiv 7 \pmod{97}$.

Pohlig-Hellman algorithm is feasible to apply only if all the q_i are small. So, for this reason, in Diffie-Hellman key exchange and Elgamal algorithm, we have to ensure that $p-1$ has a large prime factor.

Notice that, for applying Pohlig-Hellman algorithm, we need to know the prime factorisation of n, the order of the group. In the next subsection, we discuss the Baby-Step, Giant-Step algorithm due to Daniel Shanks for discrete logarithm for which we don't need the prime factorisation of n. Our discussion follows [2].

7.5.2 Baby-Step Giant-Step Algorithm

Suppose we want to solve Eqn. (16), i.e. the equation $\alpha^x \equiv \beta \pmod n$. In Baby-Step Giant-Step algorithm, we do the following:

We choose $m = \lceil \sqrt{n} \rceil$ and write $x = mq + r$ where $0 \le r < m$. The baby-step finds r and the giant-step finds q if $q \neq 0$.

Baby-Step We first compute

$$\left\{ \left(\beta\alpha^{-r}, r\right) \middle| \, 0 \le r < m \right\}. \tag{30}$$

If $(1, r)$ figures in the list for some r, say r_0, then r_0 is a solution of Eqn. (16) and we are done.

Giant-Step Otherwise, we set $\gamma = \alpha^m$ and find the values γ, γ^2, …, successively and see if any of the values match the values $\beta\alpha^{-r}$ we computed in the previous step. If the value of γ^q matches the first coordinate of one of the values in Eqn. (30), i.e. $\gamma^q = \beta\alpha^{-r}$ for some q and some r, $0 \leq r \leq m$, then $\gamma^q = \beta\alpha^{r_0}$ or $\alpha^{mq} = \beta\alpha^{-r_0}$. In other words, $x = mq + r_0$ is a solution to Eqn. (16). Note that, we can stop as soon as a value of γ^q matches the first coordinate of the values we computed in Eqn. (30).

Although we don't need the factorisation of n in this method, we need to store about $\sqrt{n}$ pairs we compute in the baby-step part of the method.

Example 11: Let us solve the equation $5^x \equiv 13 \pmod{97}$ using this method.

We have $m = \lceil\sqrt{97}\rceil = 10$, $\beta = 13$, $\alpha = 5$. So,

For $r = 0$,	$\beta\alpha^0 = 13$.	For $r = 1$,	$\beta\alpha^{-1} = 22$.
For $r = 2$,	$\beta\alpha^{-2} = 82$.	For $r = 3$,	$\beta\alpha^{-3} = 94$.
For $r = 4$,	$\beta\alpha^{-4} = 77$.	For $r = 5$,	$\beta\alpha^{-5} = 93$.
For $r = 6$,	$\beta\alpha^{-6} = 38$.	For $r = 7$,	$\beta\alpha^{-7} = 27$.
For $r = 8$,	$\beta\alpha^{-8} = 83$.	For $r = 9$,	$\beta\alpha^{-9} = 36$.

Thus, the baby step set is

$$B = \{(13,0),(22,1),(82,2),(94,3),(77,4),(93,5),(38,6),(27,7),(83,8),(36,9)\} \quad (31)$$

There is no pair in Eqn. (31) with first coordinate 1, so we carry out the giant step. We first find $\gamma = \alpha^{10} \equiv 5^{10} \equiv 53 \pmod{97}$. Since 53 is not the first coordinate of any of the pairs in Eqn. (31), we compute $53^2 \equiv 93 \pmod{97}$. We see that 93 appears as the first coordinate in the sixth element of Eqn. (31). The value of r in this case is 5 and the value of q is 2. So, $x = 2 \cdot 10 + 5 = 25$ is the solution to our equation $5^x \equiv 13 \pmod{97}$

* * *

Try the following exercise to check your understanding of the baby-step, giant-step algorithm.

E13) Solve the equation $5^x \equiv 4 \pmod{73}$ using baby-step, giant-step method.

While this algorithm has the advantage that we don't need the factorisation of the order of the group, the memory requirements become large when the order of the group is $\approx 10^{20}$. In the next subsection, we discuss the index calculus algorithm for finding discrete logarithms.

7.5.3 Index Calculus Algorithm

In this subsection, we will discuss the index calculus algorithm which is similar to the quadratic sieve algorithm. Our discussion follows [15]. The algorithm has the following steps.

Precomputation In this step we fix a bound B and suppose $q_1, q_2, \ldots, q_m$ be the primes that are less than B. We call these primes our **factor base**. We then generate relations of the form

$$\alpha^k \equiv \prod_{i=1}^{m} q_i^{r_i} \pmod{p}. \quad (32)$$

We choose a random k and store the relation if all the prime factors of α^k are less than B. For each relation of the form Eqn. (32), by taking discrete logarithm to the base α, we get a linear equation

$$k \equiv \sum_{i=1}^{m} r_i \log_\alpha q_i \pmod{p-1} \tag{33}$$

Once we have enough relations of the type Eqn. (33) we solve for $\log_\alpha q_i$.

Discrete logarithm

To solve $\alpha^x \equiv \beta \pmod{p}$ we choose r at random and find $\beta\alpha^{-r} \pmod{p}$ and see if it is B smooth, i.e. if the prime factors of $\beta\alpha^{-r}$ are less than B. If yes, we can write

$$\beta \equiv \alpha^r \prod_{j=1}^{m} q_i^{s_i} \pmod{p}$$

or

$$\log_\alpha \beta = r + \sum_{j=1}^{m} s_i \log_\alpha q_i \tag{34}$$

Since we know the values of r, s_i and $\log_\alpha q_i$, we can find $\log_\alpha \beta$.

Of course, once we do the precomputation, we can reuse it for computing several discrete logs for the same prime p.

Let us now look at an example to understand this method.

Example 12: Let $p = 137$ and $\alpha = 2$. Let $B = 10$, so we are working with the primes $2, 3, 5, 7$. A calculation yields the following:

$$5^1 \equiv 5 \pmod{137}$$
$$5^7 \equiv 5 \cdot 7 \pmod{137}$$
$$5^{50} \equiv 5 \cdot 3^3 \pmod{137}$$
$$5^{65} \equiv 2^4 \cdot 5 \pmod{137}$$

Therefore,

$$1 \equiv \log_5(5) \pmod{136} \tag{35}$$
$$7 \equiv \log_5(5) + \log_5(7) \pmod{136} \tag{36}$$
$$65 \equiv 4\log_5(2) + \log_5(5) \pmod{136} \tag{37}$$
$$50 \equiv 3\log_5(3) + \log_5(5) \pmod{136} \tag{38}$$

Note that we are abusing the notation here. We are not taking the usual logarithm with respect to 5, but taking the discrete logarithm in the group $\mathbf{Z}_{137}^*$ to the base $\overline{5}$.

Eqn. (35) and Eqn. (36) yield $\log_5(7) \equiv 6 \pmod{136}$.

Eqn. (35) and Eqn. (37) yield

$$64 = 4\log_5(2) \pmod{136} \text{ or } 64 - 4\log_5(2) = 136t$$

for some integer t. Dividing the second equation throughout by four, we get $16 - \log_5(2) = 34t$ or $16 \equiv \log_5(2) \pmod{34}$. This means that $\log_5(2)$ is congruent to

$16 \pmod{135}$ or $16+34=50 \pmod{136}$ or $50+34=84 \pmod{136}$ or $84+34=118 \pmod{136}$. We find 5^{16}, 5^{50}, 5^{84} and 5^{118} and see which of them equals $2 \pmod{137}$. We see that $5^{118} \equiv 2 \pmod{137}$, so $\log_5(2) = 118$.

From Eqn. (35) on the preceding page and Eqn. (38) on the facing page, we get $3\log_5(3) \equiv 49 \pmod{136}$. Since 3 is coprime to 136, we can solve this to get $\log_5(3) = 107$.

Suppose now that we want to find $\log_5(29)$. Finding $29 \cdot 5^{-r} \pmod{137}$ for some random values of r, we find that $29 \cdot 5^{-31} \equiv 135 = 3^3 \cdot 5 \pmod{137}$. So,

$$\log_5(29) \equiv 3\log_5(3) + \log_5(5) + 31\log_5(5) = 321 + 1 + 31 \equiv 81 \pmod{136}$$

So, $5^{81} \equiv 29 \pmod{137}$.

* * *

Here is an exercise for you to test your understanding of index calculus algorithm.

E14) Solve the equation $2^x \equiv 31 \pmod{163}$ using index calculus algorithm. You are given the following information:

$$2^{12} \equiv 21 \pmod{163}$$
$$2^{17} \equiv 20 \pmod{163}$$
$$2^{40} \equiv 9 \pmod{163}$$

Further, it is known that $31 \cdot 2^{-14} \equiv 45$.

We end this unit here. In the next section, we summarise the contents of the Unit.

7.6 SUMMARY

Let us summarise the main topics that we have discussed in this unit. In this Unit, we have discussed

1) the concept of a Public Key Cryptosystem;
2) the concept a one way function;
3) the RSA algorithm;
4) the ElGamal algorithm;
5) the Fermat factorisation algorithm, $p-1$ algorithm and the quadratic sieve algorithm for factorising integers; and
6) the Pohling-Hellman algorithm, Baby Step-Giant Step alogrithm and the index calculus algorithm for discrete logarithms.

7.7 SOLUTIONS/ANSWERS

E1) We have $\phi(n) = 6 \times 10 = 60$. However, 6 and 60 are not coprime, so we cannot choose 6 as our exponent. However, we can choose $e = 7$ since $(7, 60) = 1$. Using extended euclidean algorithm, we get

$$7 \cdot (-17) + 2 \cdot 60 = 1$$

So, we can choose $d = 43 \equiv -17 \pmod{60}$. Using repeated squaring algorithm, we have:

At the end of iteration 1, $m = 3$ $P = 40$ $b = 60$
At the end of iteration 2, $m = 1$ $P = 13$ $b = 58$
At the end of iteration 3, $m = 0$ $P = 61$ $b = 53$

So,

$$\mathscr{M}^7 \equiv 61 \pmod{77}$$

Also, check that $61^{43} \equiv 40 \pmod{77}$.

E2) Here, $\phi(391) = 16 \cdot 22 = 352$. Using extended Euclidean Algorithm, we get

$$71 \cdot 119 - 24 \cdot 352 = 1,$$

so $d = 119$. So, we have

$$c^{119} = 171^{119} \equiv 273 \pmod{391}$$

E3) From $\phi(n) = n - (p+q) + 1$, we get

$$30940 = 31309 - (p+q) + 1$$
$$\therefore p + q = 31309 - 30940 + 1 = 369$$

So, the quadratic equation is $x^2 - 370x + 31309$. We leave it to you to solve the equation and find p and q.

E4) We have

$a_1 = 78$ $N_1 = 1926049$ $N_1' = 694$
$a_2 = 254$ $N_2 = 1496357$ $N_2' = 654$
$a_3 = 66$ $N_3 = 1258997$ $N_3' = 1187$

Also, $N = 1904862461$. So,

$$\begin{aligned} x &= 78 \cdot 1926049 \cdot 694 + 254 \cdot 1496357 \cdot 654 + 66 \cdot 1258997 \cdot 1187 \\ &= 451462066854 \equiv 9663597 \pmod{1904862461} \end{aligned}$$

We have $\sqrt[3]{9663597} = 213$. Therefore, $m = 213$.

E5) Both of them will have 126 since $125^7 \equiv 28^9 \equiv 126 \pmod{127}$.

E6) In this exercise, let $O(g)$ denote the order of $g \in G$. In general, we have

$$O\left(g^k\right) = \frac{O(g)}{(k, O(g))}. \tag{39}$$

Suppose g has order n. Then, $g^n = 1$. Use Eqn. (39) to show that $g_i = g^{t_i}$ has order $q_i^{n_i}$. So, $g_i^{q_i^{n_i-1}} \neq 1$.

Conversely, suppose that $g^n = 1$ and $g_i^{q_i^{n_i-1}} \neq 1$ for each i. We have $O(g) \mid n$ since $g^n = 1$. Further, g_i has order $q_i^{n_i}$. Since $O(g_i) = O\left(g^{t_i}\right)$ and $O(g^k) \mid O(g)$ for all $k \in \mathbf{N}$, from Eqn. (39), it follows that $q^{n_i} \mid O(g)$. Since $q^{n_1}, q^{n_2}, \cdots$ are pairwise coprime, it follows that $n \mid O(g)$. Since $n \mid O(g)$ and $O(g) \mid n$, it follows that $O(g) = n$.

E7) Here $p-1 = 180 = 2^2 \cdot 3^2 \cdot 5$. Let us take $g = 2$ and check if it is a primitive root. We have $q_1 = 2$, $q_2 = 3$, $q_3 = 5$, $s_1 = 2 \cdot 3^2 \cdot 5 = 90$, $s_2 = 2^2 \cdot 3 \cdot 5 = 60$ and $s_3 = 2^2 \cdot 3^2 = 36$. . We have

$$2^{90} \equiv 180 \equiv -1 \pmod{181}$$
$$2^{60} \equiv 48 \not\equiv 1 \pmod{181}$$
$$2^{36} \equiv 59 \not\equiv 1 \pmod{181}$$

So, 2 is a primitive root mod 181.

We have $g^k = 7$. So, as in the example Bob computes $\left(g^k\right)^{p-1-x} = 7^{159} \equiv 162 \pmod{181}$. $\mathscr{M} = 122 \cdot 162 \equiv 35 \pmod{181}$. So, the message is $\mathscr{M} = 35$.

E8) We have $ed - 1 = 23936 = 2^7 \cdot 187$. We take $b = 2$. We are given that

$$b_0 \equiv 2^{187} \equiv 9079 \pmod{12193}$$
$$b_1 \equiv 9079^2 \equiv 3561 \pmod{12193}$$
$$b_2 \equiv 3561^2 \equiv 1 \pmod{12193}$$

We see that $b_1 \not\equiv \pm 1 \pmod{12193}$. So, we have $(b_1 - 1, 12193) = (3560, 12193) = 89$. We can easily check 89 is a prime by checking that none of the primes less than $\sqrt{\lfloor 89 \rfloor} = 9$, namely 2, 3, and 5, is divides 89. Dividing 12193 by 89, we get the other factor 137. We similarly check that 137 is a prime by checking that none of the primes less than $\sqrt{\lfloor 137 \rfloor} = 11$ divides 137. So, $n = 89 \cdot 137$.

E9) We have $2^{10!} \equiv 2627 \pmod{16867}$ and $(2626, 16867) = 101$. The other factor is 167.

E10) We have $51067 + 3^2 = 226^2$. Therefore, $51067 = (226+3)(226-3) = 223 \cdot 229$.

E11) Multiplying the first two relations, we get

$$(486 \cdot 972)^2 \equiv \left(2^2 \cdot 7^4\right)^2 \pmod{115697}.$$

But, we have $486 \cdot 972 \equiv 2^2 \cdot 7^4 \pmod{115697}$, so this relation gives us nothing. Multiplying the first and third relations, we get

$$(486 \cdot 1407)^2 \equiv \left(2^5 \cdot 5 \cdot 7^2\right)^2 \pmod{115697}.$$

Here, $486 \cdot 1407 \not\equiv 2^5 \cdot 5 \cdot 7^2 \pmod{115697}$. Also,

$$(486 \cdot 1407 - 2^5 \cdot 5 \cdot 7^2, 115697) = 911.$$

So, 911 is a factor of 115697. Dividing 115697 by 911, we get the other factor 127.

E12) We have $n = 96 = 2^5 \cdot 3$. So, $q_1 = 2$, $q_2 = 3$, $r_1 = 5$, $r_2 = 1$. Further, $n_1 = 3$ and $n_2 = 32$. Also, $\alpha \equiv 5 \pmod{97}$ and $\beta \equiv 7 \pmod{97}$. Let us write

$$x(2) = x_0(2) + x_1(2) \cdot 2 + x_2(2) \cdot 2^2 + x_3(2) \cdot 2^3 + x_4(2) \cdot 2^4$$

So,

$$\alpha_1 \equiv 5^3 \equiv 28 \pmod{97} \text{ and } \beta_1 \equiv 52 \pmod{97}$$

Further,

$$\gamma_1 = 28^{2^{5-1}} = 28^{16} \equiv 96 \pmod{97} \text{ and } \delta_0(2) = \beta_1^{2^{5-1}} = 52^{16} \equiv 96 \pmod{97}$$

To find $x_0(2)$, we have to solve the equation

$$96^x \equiv 96 \pmod{97}.$$

Therefore, $x_0(2) = 1$.

Next, we compute

$$\delta_1(2) = \beta_1 \alpha_1^{-x_0(2)} \equiv 52 \cdot 28^{-1} = 52 \cdot 52 \equiv 85 \pmod{97}$$

and

$$\delta_1^{2^{5-1-1}} = \delta_1^8 = 85^8 \equiv 96 \pmod{97}$$

So, again, we have to solve the equation

$$96^x \equiv 96 \pmod{97}$$

to get $x_1(2)$ and the solution is again 1.

Next, we compute

$$\delta_2(2) = \delta_1(2)\alpha^{-2 \cdot x_1(2)} = 85 \cdot 28^{-2} \equiv 47 \pmod{97}$$

To find $x_2(2)$, we have to solve the equation

$$\gamma_1^x = \delta_2(2)^{2^{5-2-1}} \text{ or } 96^x \equiv 96 \pmod{97}$$

So, $x_2(2) = 1$.

To find $x_3(2)$, we have to solve the equation $\gamma_1^x = \delta_3(2)^2$. We have

$$\delta_3(2) = \delta_2(2)\alpha^{-x_2(2)2^2} = 47 \cdot 28^{-4} \equiv 75 \pmod{97}$$

and $\delta_3(2)^2 \equiv 96 \pmod{97}$. Thus, $x_3(2)$ is a solution to $96^x \equiv 96$ and so $x_3(2) = 1$.

To find $x_4(2)$, we have to solve

$$\gamma_1^x = \delta_4(2) = \delta_3(2)\alpha_1^{-x_3(2) \cdot 2^3} = 75 \cdot 28^{-8} \equiv 96 \pmod{97}$$

Again, $x_4(2) = 1$. So, $x(2) = 31$.

We next find $x(3) = x_0(3)$. We have

$$\alpha_2 = \alpha^{n_2} = \alpha^{32} \equiv 35 \pmod{97}, \beta_2 = \beta^{n_2} \equiv 35 \pmod{97}$$

To find $x_0(3)$, we have to solve the equation

$$\gamma_2^x = \delta_0(3) \text{ where } \delta_0(3) = \beta_2 \equiv 35 \pmod{97} \text{ and } \gamma_2 = \alpha_2 \equiv 35 \pmod{97}$$

So, $x_0(3) = 1$. Solving the congruences

$$x \equiv 31 \pmod{32}$$
$$x \equiv 1 \pmod{3}$$

we get $x = 31$.

E13) The set for the baby-step is

$$B = \{(4,0),(30,1),(6,2),(45,3),(9,4),(31,5),(50,6),(10,7),(2,8),(15,9)\}. \quad (40)$$

There is no pair in Eqn. (40) with first coordinate 1. So, we carry out the giant-step. We set $\gamma = 5^9 = 10$ and we see that this occurs as the first coordinate of the eighth pair in Eqn. (40) and the value of r for this pair is 7. We have $q = 1$, so $x = 1 \cdot 9 + 7 = 16$.

E14) We have the following equations:

$$1 \equiv \log_2(2) \pmod{162} \tag{41}$$
$$12 \equiv \log_2(3) + \log_2(7) \pmod{162} \tag{42}$$
$$17 \equiv 2\log_2(2) + \log_2(5) \pmod{162} \tag{43}$$
$$40 \equiv 2\log_2(3) \pmod{162} \tag{44}$$

From Eqn. (41) and Eqn. (43), we see that $\log_2(5) = 15$.

From Eqn. (44), we see that $20 \equiv \log_2(3) \pmod{81}$. So, $\log_2(3) \equiv 20$ or $20 + 81 = 101 \pmod{162}$. Checking, we find that $2^{101} \equiv 3 \pmod{163}$, so $\log_2(3) = 101$.

Using the value of $\log_2(3)$ in Eqn. (42), we see that

$$\log_2(7) = 12 - 101 = -89 \equiv 73 \pmod{162}$$

We are given that $31 \cdot 2^{-14} \equiv 45 = 3^2 \cdot 5 \cdot \pmod{163}$. So,

$$\log_2(31) = 14 + 2\log_2(3) + \log_2(5) = 14 + 202 + 15 \equiv 69 \pmod{162}$$