MIT ART DESIGN TECHNOLOGY UNIVERSITY

SCHOOL OF ENGINEERING, LONI KALBHOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LABORATORY MANUAL

Programming Lab – VI (Information Security)

Index

Assignment No. 1

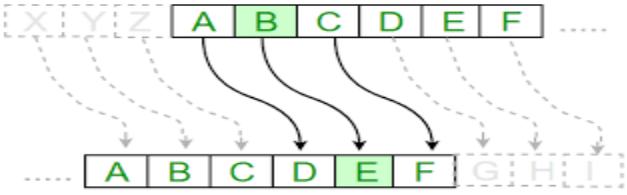
Implementation of Substitution and Transposition Ciphers

AIM: Write a program in C++ or Java to implement the following substitution and transposition ciphers.

- Caesar cipher
- Playfair cipher
- Columnar Transposition cipher.

Caesar cipher

To encrypt a message with a Caesar cipher, each letter in the message is changed using a simple rule: shift by three. Each letter is replaced by the letter three letters ahead in the alphabet. A becomes D, B becomes E, and so on. For the last letters, we can think of the alphabet as a circle and "wrap around". W becomes Z, X becomes A, Y bec mes B, and Z becomes C. To change a message back, each letter is replaced by the one three before it.



ALGORITHM:

1. Read the plain text from the user.
2. Read the key value from the user.
3. If the key is positive then encrypt the text by adding the key with each character in the plain text.
4. Else subtract the key from the plain text.
5. Display the cipher text obtained above.

Playfair cipher

The Playfair cipher starts with creating a key table. The key table is a 5×5 grid ofletters that will act as the key for encrypting your plaintext. Each of the 25 letters must be unique and one letter of the alphabet is omitted from the table (as there are 25

spots and 26 letters in the alphabet).

To encrypt a message, one would break the message into digrams (groups of 2 letters) such that, for example, "HelloWorld" becomes "HE LL OW OR LD", and map them out on the key table. The two letters of the diagram are considered as the opposite corners of a rectangle in the key table. Note the relative position of the corners of this rectangle. Thenapply the following 4 rules, in order, to each pair of letters in the plaintext:

1. If both letters are the same (or only one letter is left), add an "X" after the first letter
2. If the letters appear on the same row of your table, replace them with the letters totheir immediate right respectively
3. If the letters appear on the same column of your table, replace them with the letters immediately below respectively
4. If the letters are not on the same row or column, replace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair.

Example1: Plaintext: CRYPTO IS TOO EASY **Key =** INFOSEC **Ciphertext:** ??

Grouped text: CR YP TO IS TO XO EA SY

Ciphertext: AQ TV YB NI YB YF CB OZ

I / J	N	F	O	S
E	C	A	B	D
G	H	K	L	M
P	Q	R	T	U
V	W	X	Y	Z

ALGORITHM:

1. Read the plain text from the user.
2. Read the keyword from the user.
3. Arrange the keyword without duplicates in a 5*5 matrix in the row order and fill the remaining cells with missed out letters in alphabetical order. Note that ‘i’ and ‘j’ takes the same cell.
4. Group the plain text in pairs and match the corresponding corner letters by forming a rectangular grid.
5. Display the obtained cipher text.

Columnar Transposition cipher

Given a plain-text message and a numeric key, cipher/de-cipher the given text using Columnar Transposition Cipher. The Columnar Transposition Cipher is a form of transposition cipher just like Rail Fence Cipher. Columnar Transposition involves writing the plaintext out in rows, and then reading the ciphertext off in columns one by one.

A more complex scheme

Write letters of message out in rows over a specified number of columns. Then reorder the columns according to some key before reading off the rows

Key: 4 3 1 2 5 6 7

Plaintext: a t t a c k p

o s t p o n e

d u n t I l t

w o a m x y z

Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

- This type of transposition is not secure
- To make more secure, perform more than one transposition
- More complex permutation is not easily constructed
- E.g.re-encrupt the foregoing

Key: 4 3 1 2 5 6 7

Plaintext: t t n a a p t

m t s u o a o

d w c o i x k

n l y p e t z

Ciphertext: NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

ALGORITHM:

1. Read the Plain text.
2. Arrange the plain text in row columnar matrix format.
3. Now read the keyword depending on the number of columns of the plain text.

4. Arrange the characters of the keyword in sorted order and the corresponding columns of the plain text.
5. Read the character's row wise or column wise in the former order to get the cipher text.

INPUT **:** Plaintext

OUTPUT **:** Ciphertext

Assignment No. 2

Implementation of Diffie Hellman Key Exchange Algorithm

AIM: Write a program in C++ or Java to implement Diffie Hellman Key Exchange Algorithm.

DESCRIPTION:

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

- For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.
- P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.

Alice	Bob
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Key generated = $x = G^a mod P$	Key generated = $y = G^b mod P$
Exchange of generated keys takes place	
Key received = y	key received = x
Generated Secret Key = $k_a = y^a mod P$	Generated Secret Key = $k_b = x^b mod P$

Algebraically, it can be shown that

Ka=Kb

ALGORITHM:

1. Both Alice and Bob share the same public keys g and p.
2. Alice selects a random public key a.
3. Alice computes his secret key A as g^a mod p.
4. Then Alice sends A to Bob.
5. Similarly, Bob also selects a public key b and computes his secret key as Band sends the same back to Alice.
6. Now both of them compute their common secret key as the other one's secretkey power of a mod p.

EXAMPLE:

1. Alice and Bob get public numbers P = 23, G = 9
2. Alice selected a private key a = 4 and
   Bob selected a private key b = 3
3. Alice and Bob compute public values
   Alice: x = (9^4 mod 23) = (6561 mod 23) = 6
   Bob: y = (9^3 mod 23) = (729 mod 23) = 16
4. Alice and Bob exchange public numbers
5. Alice receives public key y =16 and
   Bob receives public key x = 6
6. Alice and Bob compute symmetric keys
   Alice: ka = y^a mod p = 65536 mod 23 = 9
   Bob: kb = x^b mod p = 216 mod 23 = 9
7. 9 is the shared secret.

INPUT **:** P and g

OUTPUT **:** Shared secret key

Assignment No. 3

Implementation of RSA

AIM: Write program in C++ or Java to implement RSA algorithm for key generation and cipher verification.

DESCRIPTION:

Asymmetric/Public Key Algorithm:

Public key algorithms were evolved to solve the problem of key distribution in symmetric algorithms. This is achieved by using one key for encryption and a different but related key for decryption. These algorithms are designed such that it is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key. Also in some algorithms, such as RSA, either of the two related keys can be used for encryption, with the other used for decryption.

A public key encryption scheme has six ingredients:

- Plaintext: This is readable message or data that is given as input to the algorithm.
- Encryption algorithm: The algorithm that transforms the plaintext into cipher text.
- Public and private key: This is a pair of keys generated such that if one is used for encryption, the other is used for decryption.
- Cipher text: This is the encoded message produced as output of encryption. It depends on the plaintext and the key.
- Decryption algorithm: The algorithm that transforms the cipher text into plain text with the help of key.

The essential steps in public key algorithm are as follows:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or the other accessible file. This is the public key. The companion key is kept private. Each user maintains a collection of public keys obtained from other parties participating in communication.
3. If A wishes to send a confidential message to B, A encrypts the message using B's public key.
4. When B receives the message, B decrypts it using B's private key. No other recipient can decrypt the message because only B knows B's private key.

RSA Algorithm

RSA (which stands for Rivest, Shamir and Adleman who first publicly described it), an algorithm for public-key cryptography involves three steps key generation, encryption and decryption.

RSA is a block cipher with each block having a binary value less than some number n. That is the block size must be less than or equal to log2 (n). Encryption & decryption are of the following form,

for some plaintext block M and cipher text block C:

$C = M^e \bmod n$

$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$

Both sender and receiver must know the value of n. The sender knows the value of e and only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of PU = {e, n} and a private key of PR = {d, n}. For this algorithm to be satisfactory for public key encryption, the following requirements must meet:

1. It is possible to find values of e, d, n such that $M^{ed} = M \bmod n$ for all M<n.
2. It is relatively easy to calculate M^e and C^d for all values of M<n.
3. It is infeasible to determine d given e and n.

ALGORITHM:

a. Key generation

The keys (public key and private key) for RSA algorithm are generated as:

1. Choose two distinct prime numbers p and q.

   For security purposes, the integer p and q should be chosen at random, and should be of similar bit-length.
2. Compute n = pq.

   n is used as the modulus for both the public and private keys
3. Compute $\varphi(n) = (p-1)(q-1)$, where φ is Euler's totient function
4. Choose an integer e such that

   $1 < e < \varphi(n)$

   gcd $(e, \varphi(n)) = 1$, i.e. e and $\varphi(n)$ are co prime.

   - e is released as the public key exponent.
   - e having a short bit-length and small Hamming weight results in more efficient encryption - most commonly 0x10001 = 65537. However, small values of e (such as 3) have been shown to be less secure in some settings.

5. Determine d that is the multiplicative inverse of e mod $\varphi(n)$.
   - This is more clearly stated as solve for d given $(d*e) \bmod \varphi(n) = 1$
   - This is often computed using the extended Euclidean algorithm.
   - d is kept as the private key exponent.

   Public key : PU = {e, n}
   Private Key : PR = {d, n}

b. Encryption

Alice transmits her public key (e, n) to Bob and keeps the private key secret. Bob then wishes to send

message M to Alice. He computes the cipher text c corresponding to

$$C = M^e \bmod n$$

This can be done quickly using the method of exponentiation by squaring. Bob then transmits C to Alice.

c. Decryption

Alice can recover M from C by using her private key exponent d via computing

$$M = C^d \bmod n$$

EXAMPLE:

1. Select two prime numbers, p = 17 and q = 11.
2. Calculate n = pq = 17*11 = 187.
3. Calculate Ø(n) = (p-1)(q-1) = 16*10 = 160.
4. Select e such that relatively prime to Ø(n) = 160 and less than Ø(n); we choose e = 7.
5. Determine d such that de ≡ 1 (mod 160) and d < 160. The correct value is d = 23, because 23*7 = 161 = 10*160+1; d can be calculated using the extended Euclid's algorithm.

The resulting keys are public key PU = {7, 187} and private key PR = {23, 187}. The example shows the use of these keys for plaintext input of M=88.

INPUT **:** P, Q and Plaintext

OUTPUT **:** Ciphertext

Assignment No. 4

Implementation of Chinese Remainder Theorem

AIM: Write a program in C++ or Java based on number theory such as Chinese remainder.

DESCRIPTION:

Relative Prime Numbers

Two integers are termed relative prime if the only common factor between them is 1.

i.e., Greatest Common Divisor (m, n) = 1

Any integer can be broken down into certain multiples of prime numbers. This is called prime factorization. When two integers are prime factorized and the only common number is 1, then the two integers are relatively prime.

Two distinct primes and are always relatively prime.

Example

18 = 2 x 3 x 3

35 = 7 x 5

Here 18 and 35 are relative prime as there is no common factor between them.

18 = 2 x 3 x 3

21 = 3 x 7

Here 18 and 21 are not relatively prime as 3 is a common factor between them.

Set of Residues

It is a set of nonnegative integers less than n.

Zn = {0, 1, 2,…….(n – 1) }

Chinese Remainder Theorem (CRT)

Let m1, m2, …, mk be pair wise relatively prime positive integers. That is,

GCD (mi, mj) = 1 for $1 \leq i < j \leq k$.

Let a_i € Zm_i for $1 \leq i \leq k$

set M=m1m2…mk.

Then there exists a unique A € Zm such that

a_i = A mod m_i for i = 1……k

A can be computed as:

$$A \equiv (\sum_{i=1}^{k} a_i c_i) \bmod M$$

where $c_i = M_i \times (M_i^{-1} \bmod m_i)$ & $M_i = M / m_i$ for $1 \leq i \leq k$.

Steps in Chinese Remainder Theorem

1. Find M = m1 × m2 × … × mk.
2. Find M1 = M / m1, M2 = M / m2… Mk = M / mk.
3. Find the multiplicative inverse of M1, M2, …, Mk using the corresponding moduli.
   (m1, m2, …, mk). Call the inverses $M1^{-1}$, $M2^{-1}$, …, Mk^{-1}.
4. The solution to the simultaneous equations is

$$x = (a_1 \times M_1 \times M_1^{-1} + a_2 \times M_2 \times M_2^{-1} + \cdots + a_k \times M_k \times M_k^{-1}) \bmod M$$

Example1

Represent 973 in Z_{1813} as a k-tuple.

Answer:

- M = 1813 = 37 * 49 i.e. m1 = 37 & m2 = 49
- A = 973
- A = (A mod m1, A mod m2) = (11, 42)

Example2

Find x for the following equations:

x ≡ 2 mod 3

x ≡ 3 mod 5

x ≡ 2 mod 7

Answer

1. M = 3 × 5 × 7 = 105
2. M1 = 105 / 3 = 35, M2 = 105 / 5 = 21, M3 = 105 / 7 = 15
3. The inverses are $M1^{-1} = 2$, $M2^{-1} = 1$, $M3^{-1} = 1$
4. x = (2 × 35 × 2 + 3 × 21 × 1 + 2 × 15 × 1) mod 105
   = 23 mod 105
5. x = 23

INPUT : Values of a_i and m_i

OUTPUT : Unique value of X

Assignment No. 5

Implementation of DES

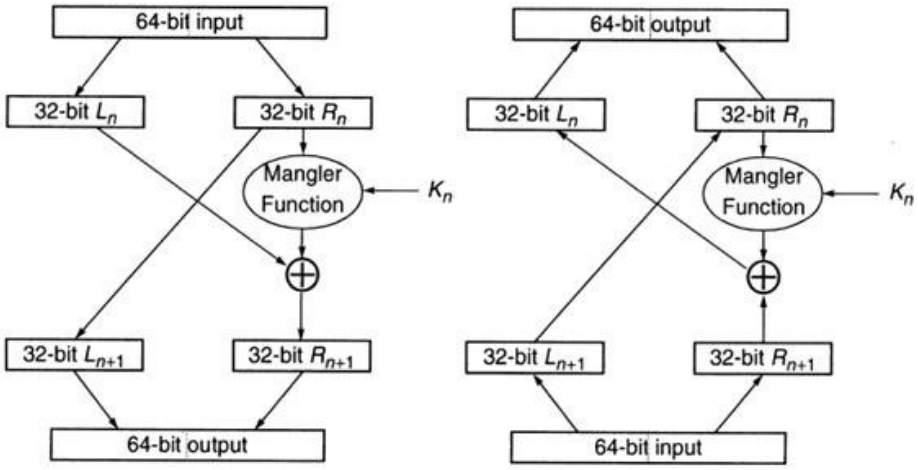
AIM: Write program in Java to implement DES algorithm using Libraries (API).

DESCRIPTION:

DES is a symmetric encryption system that uses 64-bit blocks, 8 bits of which areused for parity checks. The key therefore has a "useful" length of 56 bits, which means thatonly 56 bits are actually used in the algorithm. The algorithm involves carrying out combinations, substitutions and permutations between the text to be encrypted and the key, while making sure the operations can be performed in both directions. The key is ciphered on64 bits and made of 16 blocks of 4 bits, generally denoted k_1 to k_{16}. Given that "only" 56 bits are actually used for encrypting, there can be 2^{56} different keys.

The main parts of the algorithm are as follows:

- Fractioning of the text into 64-bit blocks
- Initial permutation of blocks
- Breakdown of the blocks into two parts: left and right, named L and R
- Permutation and substitution steps repeated 16 times
- Re-joining of the left and right parts then inverse initial permutation



ALGORITHM:

1. Read the 64-bit plain text.
2. Split it into two 32-bit blocks and store it in two different arrays.
3. Perform XOR operation between these two arrays.
4. The output obtained is stored as the second 32-bit sequence and the original second 32-bit sequence forms the first part.
5. Thus the encrypted 64-bit cipher text is obtained in this way. Repeat the same process for the remaining plain text characters.

INPUT **:** Plaintext

OUTPUT **:** Ciphertext

Assignment No. 6

Implementation of SHA-1

AIM: Write a program in Java to implement SHA-1 algorithm using Libraries (API).

DESCRIPTION:

The National Institute of Standard and Technology (NIST) along with NSA developed the Secure Hash Algorithm (SHA).SHA works with any input message that is less than 2^{64} bits in length. The output of SHA-1 is a message digest which is 160bits in length. Following table shows SHA parameters.

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

Comparison of SHA parameters

The word secure is SHA was decided based on two features. SHA is designed to be computationally infeasible to:

1. Obtain the original message, given its message digest.
2. Find two messages producing the same message digest.

Steps of SHA

1. **Padding**

The first step in SHA is to add padding to the end of the original message in such a way that the length of the message is 64 bits short of multiple of 512.Padding is always added even if message is already 64 bit short of multiple of 512.

2. **Append length**

The length of the message excluding the length of the padding is now calculated and appended to the end of the padding as a 64 bit block.

3. **Divide the input into 512 bit block**

The input message is now divided into blocks, each of length 512 bits. These blocks become the input to the message digest processing logic.

4. **Initialize chaining variables**

Five chaining variables A through E are initializes. Each chaining variable is 32 bits in length. Following table shows their values

Variable Name	Value(in Hexadecimal)
A	01 23 45 67
B	89 AB CD EF
C	FE DC BA 98
D	76 54 32 10
E	C3 D2 E1 F0

1. **Process Block**

This process is divided into following sub steps

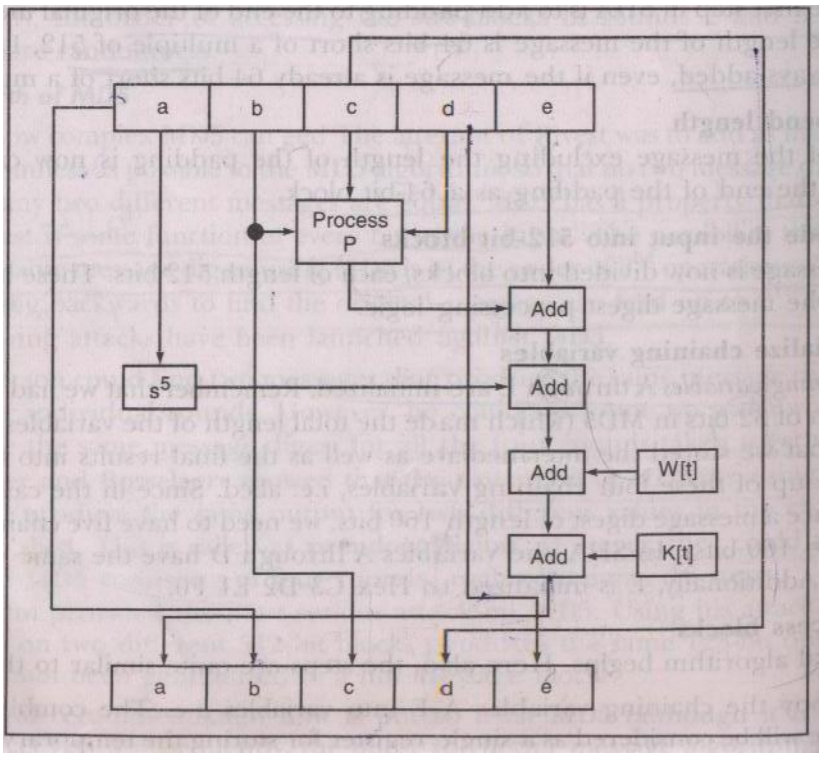
5.1 Copy the chaining variables –E into variables a-e. The combination of a-e called abcde will be considered as a single register for storing the temporary intermediate as well as the final results.

5.2 Now divide the current 512 bit block into 16 sub blocks, each consisting of 32 bits.

5.3 SHA has 4 rounds each consisting of 20 steps or iteration. Each round takes the current 512 bit block, the register abcde and a constant K[t] (where t=0 to 79) as the three inputs. It then updates the contents of the register abcde using SHA algorithm steps.We have only four constants defined for K[t],one used in each of four rounds .The values of K[t] are given in following table

Round	Value of t between	K[t](in Hexadecimal)
1	1 and 19	5A 92 79 99
2	20 and 39	6E D9 EB A1
3	40 and 59	9F 1B BC DC
4	60 and 79	CA 62 C1 D6

5.4 SHA consist of four rounds, each round containing 20 iterations. This makes a total of 80 iterations. The logical operation of SHA-1 is shown in following fig.



Single SHA-1 iteration

Mathematically iteration consists of following operations

abcde =(e + Process P +s^5 (a) + W[t] + K[t]),a,s^{30} (b),c, d

where

abcde	:The register made up of five variables a, b, c, d, e
Process P	: The logical operation given in the following table
s^t	:Circular left shift of 32 bit sub block by t bits.
W[t]	: A 32 bit value derived from current 32 bit sub block
K[t]	: one of the constant defined earlier

Process P is given as

Round	Process P
1	(b AND c) OR ((NOT b) AND (d))
2	b XOR c XOR d
3	(b AND c) OR (b AND d) OR (c AND d)
4	B XOR c XOR d

Value of W[t] is calculated as follow

For the first 16 blocks of W(i.e t=0 to 15) the contents of the inpu message sub block M[t] become the content of W[t].

Remaining values are calculated as

For t=0 to 15	Value of W[t]
W[t]=	Same as M[t]
W[t]=	(W[t -16] **XOR** W[t-14] **XOR** W[t-8] **XOR** W[t-3])

Required Classes

1. **Class MessageDigest (java.security.MessageDigest)**

This MessageDigest class provides applications the functionality of a message digest algorithm, such as MD5 or SHA. Message digests are secure one-way hash functions that take arbitrary-sized data and output a fixed-length hash value.

Constructor Summary	
protected	**MessageDigest(**String algorithm**)** Creates a message digest with the specified algorithm name.
Required Methods	
static MessageDigest	**getInstance**(String algorithm) Generates a MessageDigest object that implements the specified digest algorithm.
static MessageDigest	**getInstance**(String algorithm,Provider provider) Generates a MessageDigest object implementing the specified algorithm, as supplied from the specified provider, if such an algorithm is available from the provider.
static MessageDigest	**getInstance**(String algorithm, String provider) Generates a MessageDigest object implementing the specified algorithm, as supplied from the specified provider, if such an algorithm is available from the provider.
void	**update**(byte[] input) Updates the digest using the specified byte.
void	**update**(byte[] input) Updates the digest using the specified array of bytes.
void	**update**(byte[] input, int offset, int len) Updates the digest using the specified array of bytes, starting at the specified offset.

Algorithm

User A:

1. Read the public key of Server using ObjectInputStream class.
2. Use readObject method to read an object.
3. Input the plaintext to be sent.

Generating Digest

1. Gets Instance of MessageDigest class for SHA-1 algorithm using getInstance method of MessageDigest class.
2. Convert the plaintext into bytes and pass it to update () method of MessageDigest class.
3. Use digest () method of MessageDiegest class to generate digest which returns the result in bytes format.

Encryption

1. Initialize the Cipher class object in encrypt mode.
2. Use the public key of server to encrypt the plaintext.

Sending the Request

1. Send the cipher and digest to the server using ObjectOutputStream class.
2. Use writeObject method to write both objects.

User B:

Key Generation

1. Use RSA algorithm for generation of Keys.
2. Get object of KeyPairGenerator class.
3. Initialize KeyPairGenerator.
4. Generate the pair of keys by using KeyPair class
5. Use ObjectOutputStream class & Socket programming to send the public key to client (Announce the public key).

Read the Request

1. Use ObjectInputStream class to read the cipher text and digest sent by Client. Received digest - MD1
2. Initialize object of Cipher class in Decrypt Mode.
3. Decrypt the cipher text using Private Key of Server & get the original plaintext.

Verify the digest

1. Use MessageDigest class for generating the digest using decrypted plaintext.
2. Pass the plaintext to update () method of MessageDigest class.
3. Use digest () method of MessageDiegest class to generate digest which returns byte []. Calculated digest - MD2
4. Compare received digest (MD1) with calculated digest (MD2).

5. If they are equal no modifications are done to message so accept the message. Otherwise integrity is violated so reject the message.

INPUT **:** Plaintext

OUTPUT **:** Message Digest