

## **CODE:**

```
// A C++ program to demonstrate working of Chinese remainder
// Theorem
#include<iostream.h>
using namespace std;

// k is the size of num[] and rem[]. Returns the smallest
// number x such that:
// x % num[0] = rem[0],
// x % num[1] = rem[1],
// .....
// x % num[k-2] = rem[k-1]
// Assumption: Numbers in num[] are pairwise coprime
// (gcd for every pair is 1)
int findMinX(int num[], int rem[], int k)
{
    int x = 1; // Initialize result

    // As per the Chinese remainder theorem,
    // this loop will always break.
    while (true)
    {
        // Check if remainder of x % num[j] is
        // rem[j] or not (for all j from 0 to k-1)
        int j;
        for (j=0; j<k; j++)
            if (x%num[j] != rem[j])
                break;

        // If all remainders matched, we found x
        if (j == k)
            return x;

        // Else try next number
        x++;
    }

    return x;
}
```

```

// Driver method
int main(void)
{
    int num[] = {3, 4, 5};
    int rem[] = {2, 3, 1};
    int k = sizeof(num)/sizeof(num[0]);
    cout << "x is " << findMinX(num, rem, k);
    return 0;
}

```

```

1 //C++ program to demonstrate working of Chinese remainder Theorem
2 #include<iostream>
3 using namespace std;
4
5 int findMinX(int num[], int rem[], int k)
6 {
7     int x = 1; // Initialize result
8     while (true)
9     {
10         int j;
11         for (j=0; j<k; j++)
12             if (x%num[j] != rem[j])
13                 break;
14         if (j == k)
15             return x;
16         x++;
17     }
18     return x;
19 }
20
21 // Driver method
22 int main(void)
23 {
24     int num[] = {3, 4, 5};
25     int rem[] = {2, 3, 1};
26     int k = sizeof(num)/sizeof(num[0]);
27     cout << "x is " << findMinX(num, rem, k);
28     return 0;
29 }
30

```

x is 11

**CODE:**

// C program for RSA asymmetric cryptographic

```
#include<stdio.h>
```

```
#include<math.h>
```

```
int gcd(int a, int h)
```

```
{  
    int temp;  
    while (1)  
    {  
        temp = a%h;  
        if (temp == 0)  
            return h;  
        a = h;  
        h = temp;  
    }  
}
```

// Code to demonstrate RSA algorithm

```
int main()
```

```
{  
    // Two random prime numbers  
    double p = 3;  
    double q = 7;  
  
    // First part of public key:  
    double n = p*q;  
  
    // Finding other part of public key.  
    double e = 2;  
    double phi = (p-1)*(q-1);  
    while (e < phi)  
    {  
        if (gcd(e, phi)==1)  
            break;  
        else  
            e++;  
    }  
}
```

```

    int k = 2;
    double d = (1 + (k*phi))/e;
    double msg = 20;
    printf("Message data = %lf", msg);
    double c = pow(msg, e);
    c = fmod(c, n);
    printf("\nEncrypted data = %lf", c);
    double m = pow(c, d);
    m = fmod(m, n);
    printf("\nOriginal Message Sent = %lf", m);

    return 0;
}

```

The screenshot shows a C program for RSA asymmetric cryptography. The code defines a gcd function, sets prime numbers p=3 and q=7, calculates n=p\*q, and finds a public key e=2. It then demonstrates the encryption of the message data 20.000000 to encrypted data 20.000000, and finally prints the original message sent as 20.000000.

```

1 // C program for RSA asymmetric cryptographic
2
3 #include<stdio.h>
4 #include<math.h>
5 int gcd(int a, int h)
6 {
7     int temp;
8     while (1)
9     {
10         temp = a%h;
11         if (temp == 0)
12             return h;
13         a = h;
14         h = temp;
15     }
16 }
17
18 // Code to demonstrate RSA algorithm
19 int main()
20 {
21     // Two random prime numbers
22     double p = 3;
23     double q = 7;
24
25     // First part of public key:
26     double n = p*q;
27
28     // Finding other part of public key.
29     double e = 2;
30     double phi = (p-1)*(q-1);
31     while (e < phi)
32     {

```

Message data = 20.000000  
Encrypted data = 20.000000  
Original Message Sent = 20.000000

## **CODE:**

```
/* the Diffie-Hellman Key exchange algorithm using C++ */
#include <cmath>
#include <iostream>
using namespace std;
long long int power(long long int a, long long int b,
long long int P)
{
if (b == 1)
return a;
else
return (((long long int)pow(a, b)) % P);
}
// Driver program
int main()
{
long long int P, G, x, a, y, b, ka, kb;
P = 23;
cout << "The value of P : " << P << endl;
G = 9;
cout << "The value of G : " << G << endl;
a = 4;
cout << "The private key a for Preeti : " << a << endl;
x = power(G, a, P);
b = 3;
cout << "The private key b for Kriti : " << b << endl;
y = power(G, b, P);
ka = power(y, a, P);
kb = power(x, b, P);
cout << "Secret key for the Preeti is : " << ka << endl;
cout << "Secret key for the Kriti is : " << kb << endl;
return 0;
}
```

```

5 long long int power(long long int a, long long int b,
6                     long long int P)
7 {
8     if (b == 1)
9         return a;
10
11     else
12         return (((long long int)pow(a, b)) % P);
13 }
14
15 // Driver program
16 int main()
17 {
18     long long int P, G, x, a, y, b, ka, kb;
19     P = 23;
20     cout << "The value of P : " << P << endl;
21
22     G = 9;
23     cout << "The value of G : " << G << endl;
24     a = 4;
25     cout << "The private key a for Preeti : " << a << endl;
26
27     x = power(G, a, P);
28     b = 3;
29     cout << "The private key b for Kriti : " << b << endl;
30     y = power(G, b, P);
31     ka = power(y, a, P);
32     kb = power(x, b, P);
33     cout << "Secret key for the Preeti is : " << ka << endl;
34
35     cout << "Secret key for the Kriti is : " << kb << endl;
36

```

```

The value of P : 23
The value of G : 9
The private key a for Preeti : 4
The private key b for Kriti : 3
Secret key for the Preeti is : 9
Secret key for the Kriti is : 9

```

**Code:**

```
// Java program to calculate SHA-1 hash value

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class Main {

    public static String encryptThisString(String input)

    {

        try {

            // getInstance() method is called with algorithm SHA-1

            MessageDigest md = MessageDigest.getInstance("SHA-1");

            // digest() method is called

            // to calculate message digest of the input string

            // returned as array of byte

            byte[] messageDigest = md.digest(input.getBytes());

            // Convert byte array into signum representation

            BigInteger no = new BigInteger(1, messageDigest);

            // Convert message digest into hex value

            String hashtext = no.toString(16);

            // Add preceding 0s to make it 32 bit

            while (hashtext.length() < 32) {

                hashtext = "0" + hashtext;

            }

            // return the HashText

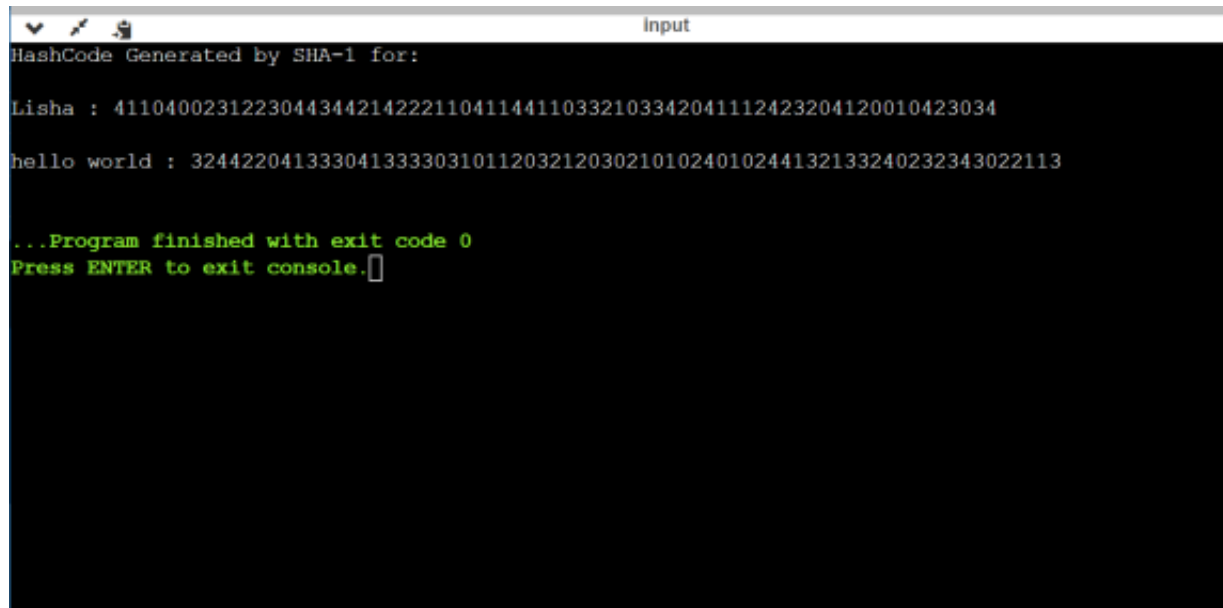
            return hashtext;

        }

        // For specifying wrong message digest algorithms

        catch (NoSuchAlgorithmException e) {
```

```
throw new RuntimeException(e);
}
}
// Driver code
public static void main(String args[]) throws
NoSuchAlgorithmException
{
    System.out.println("HashCode Generated by SHA-1 for: ");
    String s1 = "Kamal";
    System.out.println("\n" + s1 + " : " + encryptThisString(s1));
    String s2 = "hello world";
    System.out.println("\n" + s2 + " : " + encryptThisString(s2));
}
}
```

A screenshot of a Java IDE's console window. The window has a title bar with standard icons and the word "input". The console output shows the program's execution: it prints "HashCode Generated by SHA-1 for:", then "Lisha : 411040023122304434421422211041144110332103342041112423204120010423034", then "hello world : 32442204133304133330310112032120302101024010244132133240232343022113", and finally "...Program finished with exit code 0" and "Press ENTER to exit console." with a cursor.

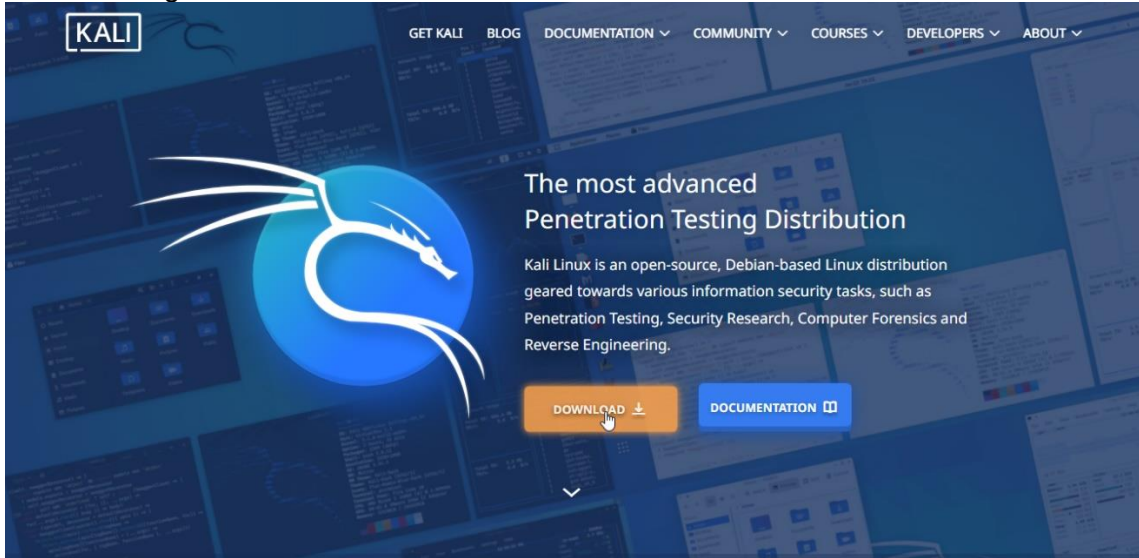
```
HashCode Generated by SHA-1 for:
Lisha : 411040023122304434421422211041144110332103342041112423204120010423034
hello world : 32442204133304133330310112032120302101024010244132133240232343022113
...Program finished with exit code 0
Press ENTER to exit console.[]
```



## Output:-

### Step 1: Download the iso file

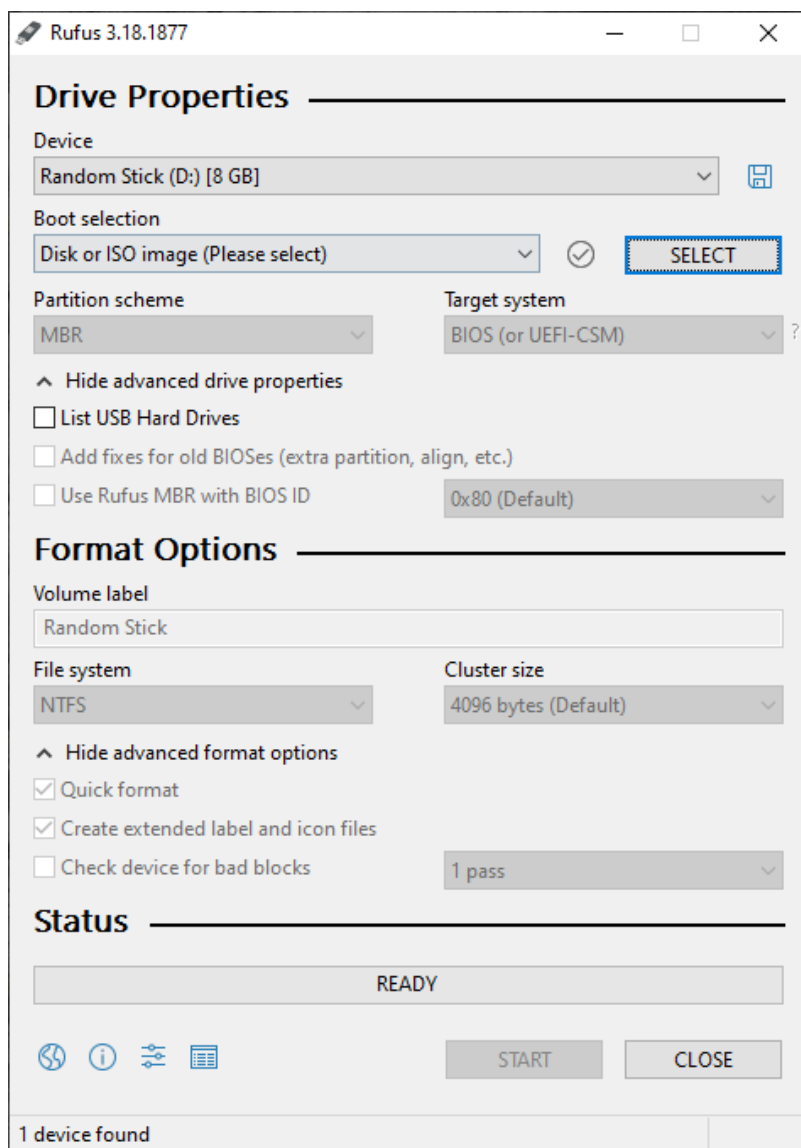
Go to kali.org and hit the download button.



### Step 2: Create a bootable drive

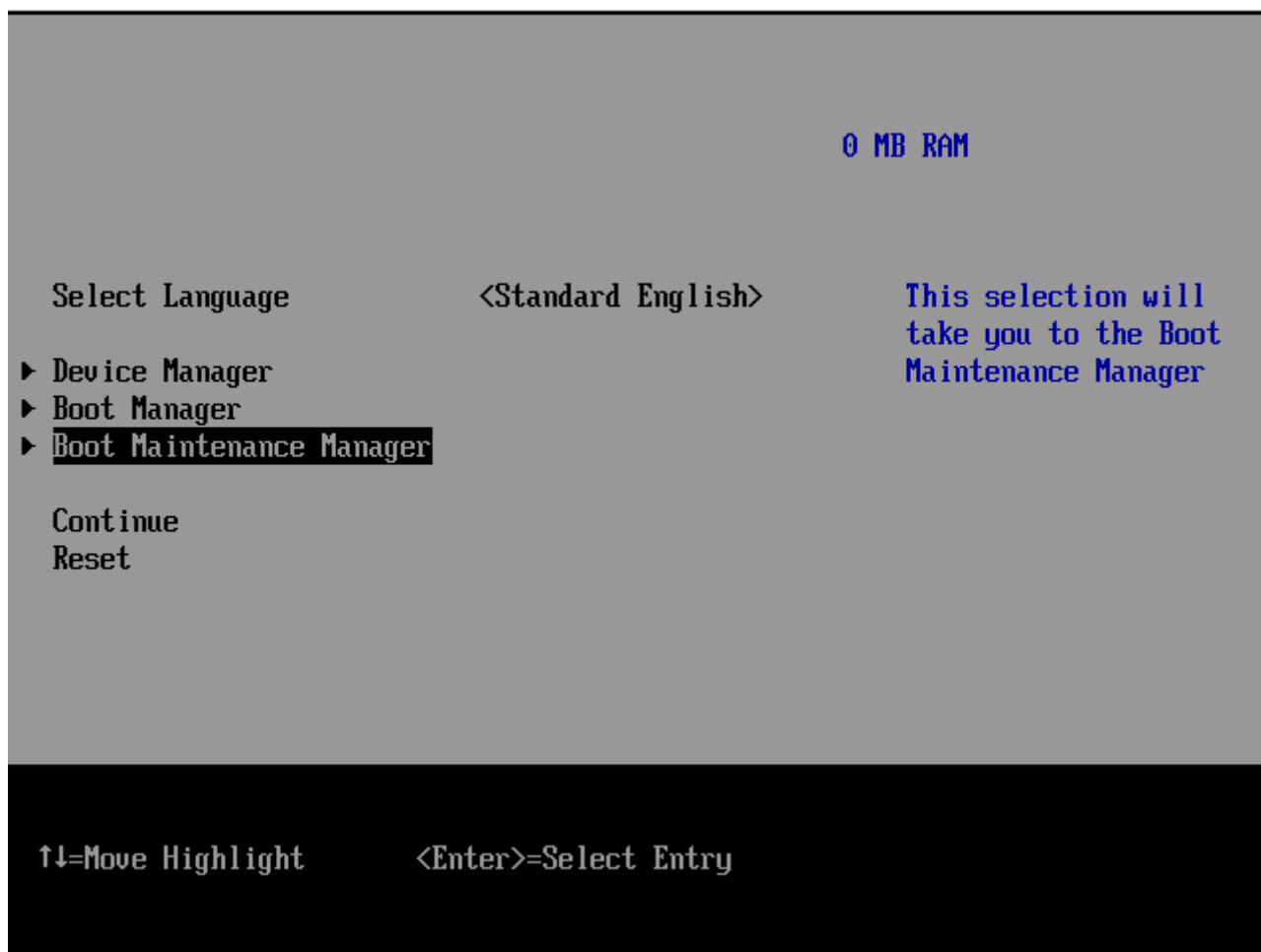
You can download Rufus from [rufus.ie](https://rufus.ie) (Rufus 3.18 as at the time of writing). In order to make the stick bootable, we are going to run Rufus and make a few changes.

Connect the stick and select it under the 'Device' options. Under 'Boot selection' select your newly downloaded Kali iso file. Now for the tricky part.



### Step 3: Access the Kali Installer Menu

To boot the computer from the new Kali USB stick, you'll need to disable secure boot if it is enabled in the BIOS settings.



#### Step 4: Begin the installation

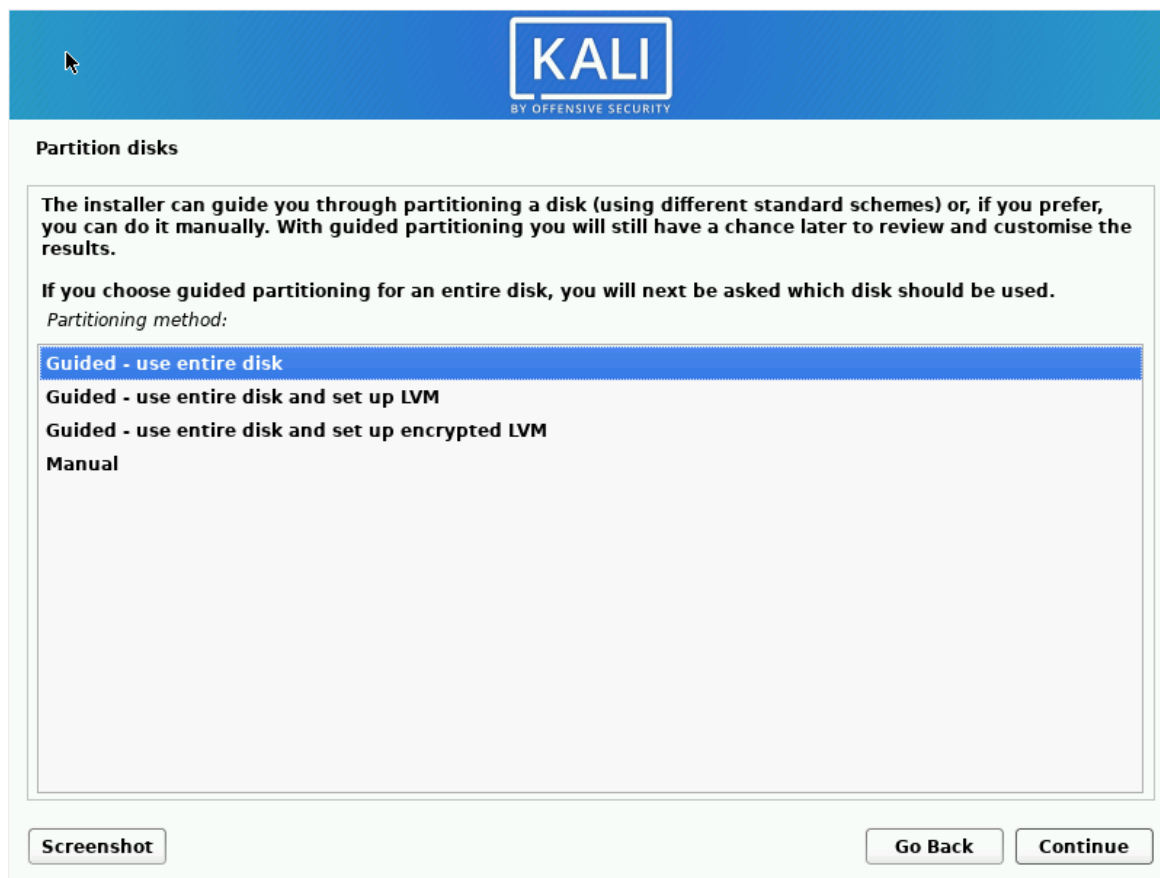
Select graphical install, and you can now use your mouse. Select your preferred language, region, and keyboard layout in the following menus:

#### Step 5: Set up the storage

Next would be to select the partitioning method. Now for the cool head mentioned earlier. If you want to format the entire hard drive for Kali, the Guided options will be best.

LVM (Logic Volume Management) is a feature that allows you to have relatively flexible partitions. This means that you can extend, shrink or even merge partitions while the OS is being run. It's a pretty nifty feature.

The encrypted LVM feature keeps your data safe if someone unauthorized gets access to your hard drive. Just note that there is a trade-off here: your hard drive will tend to be slower than if it wasn't encrypted. So most people go with the 'Guided -use entire disk' option.



### Step 5: Chose software and a desktop look

Now, choose the software you wish to install. Check the desktop environment and collection of tools options, as these will help you avoid having to install a lot of things later.

Desktop environments are basically the way the desktop looks to the user. Kali offers Xfce (most common), Gnome, and KDE. I'm a sucker for Gnome so I went with that option. You can still install all three and later configure your computer to choose the one you'd like.



## Software selection

At the moment, only the core of the system is installed. The default selections below will install Kali Linux with its standard desktop environment and the default tools.

You can customize it by choosing a different desktop environment or a different collection of tools.

*Choose software to install:*

- ☒ Desktop environment [selecting this item has no effect]
- ☐ ... Xfce (Kali's default desktop environment)
- ☒ ... GNOME
- ☐ ... KDE Plasma
- ☒ Collection of tools [selecting this item has no effect]
- ☒ ... top10 -- the 10 most popular tools
- ☐ ... default -- recommended tools (available in the live system)

Screenshot

Continue