SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(IoT and Cybersecurity Including BCT)

COURSE NAME : Fundamentals Of Cryptography

II YEAR / III SEMESTER

Unit I

Topic : Euclidean and Extended Algorithm

The Euclidean algorithm is a way to find the greatest common divisor of two positive integers. GCD of two numbers is the largest number that divides both of them. A simple way to find GCD is to factorize both numbers and multiply common prime factors.

$$36 = 2 \times 2 \times 3 \times 3$$
$$60 = 2 \times 2 \times 3 \times 5$$

$$\begin{aligned} GCD &= \text{Multiplication of common factors} \\ &= 2 \times 2 \times 3 \\ &= 12 \end{aligned}$$

Basic Euclidean Algorithm for GCD:

The algorithm is based on the below facts.

- If we subtract a smaller number from a larger one (we reduce a larger number), GCD doesn't change. So if we keep subtracting repeatedly the larger of two, we end up with GCD.
- Now instead of subtraction, if we divide the smaller number, the algorithm stops when we find the remainder 0.

Below is a recursive function to evaluate gcd using Euclid's algorithm:

```
// C program to demonstrate Basic Euclidean Algorithm
#include <stdio.h>

// Function to return gcd of a and b
int gcd(int a, int b)
{
    if (a == 0)
        return b;
    return gcd(b % a, a);
}

// Driver code
int main()
{
    int a = 10, b = 15;

      // Function call
    printf("GCD(%d, %d) = %d\n", a, b, gcd(a, b));
    a = 35, b = 10;
    printf("GCD(%d, %d) = %d\n", a, b, gcd(a, b));
    a = 31, b = 2;
    printf("GCD(%d, %d) = %d\n", a, b, gcd(a, b));
    return 0;
}
```

`GCD(10, 15) = 5 GCD(35, 10) = 5 GCD(31, 2) = 1`

Time Complexity: O(Log min(a, b))

Auxiliary Space: O(Log (min(a,b))

Extended Euclidean Algorithm:

Extended Euclidean algorithm also finds integer coefficients x and y such that: ax + by = gcd(a, b)

Examples:

Input: a = 30, b = 20
Output: gcd = 10, x = 1, y = -1
(Note that 30*1 + 20*(-1) = 10)
Input: a = 35, b = 15
Output: gcd = 5, x = 1, y = -2
(Note that 35*1 + 15*(-2) = 5)

The extended Euclidean algorithm updates the results of gcd(a, b) using the results calculated by the recursive call gcd(b%a, a). Let values of x and y calculated by the recursive call be x_1 and y_1. x and y are updated using the below expressions.

ax + by = gcd(a, b)
gcd(a, b) = gcd(b%a, a)
gcd(b%a, a) = $(b\%a)x_1 + ay_1$
ax + by = $(b\%a)x_1 + ay_1$
ax + by = $(b – [b/a] * a)x_1 + ay_1$
ax + by = $a(y_1 – [b/a] * x_1) + bx_1$
Comparing LHS and RHS,
x = y_1 – ?b/a? * x_1
y = x_1

```
/ C++ program to demonstrate working of
// extended Euclidean Algorithm
#include <bits/stdc++.h>
using namespace std;

// Function for extended Euclidean Algorithm
int gcdExtended(int a, int b, int *x, int *y)
{
    // Base Case
    if (a == 0)
    {
        *x = 0;
        *y = 1;
        return b;
    }

    int x1, y1; // To store results of recursive call
    int gcd = gcdExtended(b%a, a, &x1, &y1);

    // Update x and y using results of
    // recursive call
    *x = y1 - (b/a) * x1;
    *y = x1;

    return gcd;
}

// Driver Code
int main()
{
    int x, y, a = 35, b = 15;
    int g = gcdExtended(a, b, &x, &y);
    cout << "GCD(" << a << ", " << b
         << ") = " << g << endl;
    return 0;
```