# Homework 4

SJSU Students

May 10, 2006

## 1 Problem 31.1-7

For any integer $k > 0$, we say that an integer n is a $k^{th}$ power if there exists an integer a such that $a^k =$ n. We say that $n > 1$ is a nontrivial power if it is a $k^{th}$ power for some integer $k > 1$. Show how to determine if a given $\beta$-bit integer n is a nontrivial power in time polynomial in $\beta$.

The goal is to check if a given $\beta$-bit integer n has any root. For this we need to check for square roots, cube roots, $\cdots$, up to log n roots.

Why up to log n ?

According to the problem, n is an integer which is $n > 1$ and k is an integer which is $k > 1$. Thus the smallest possible base is 2 satisfying above problem assumption. Then the largest possible power k is of the smallest base 2. That is log n.

What to do for a given $k^{th}$ root?

First, to check if a number had a square root. keep trying to compute $2^i$ for larger and larger values of I until we found i such that $(2^i)^2 < n < (2^{i+1})^2$ then we would check if $(2^i + 2^{i-1})^2$ was greater or less than n, and in this way binary search for the largest integer whose square was less than or equal to n. If it turned out to be exactly equal then we know it has a square root. Binary search time complexity is O(log n).

Then we can use the same algorithm for checking if a cube root,$\cdots$,log n root exists. Checking for all of this in total will be O(log n * log n) = O(n).

Using this algorithm we can say this problem is solvable in polynomial time because the number of steps required to complete the algorithm for a given $\beta$-bit input n is O(n). And for each of steps need $\Theta(\beta)$ bit operations. n or $\beta$ is the complexity of the input.

# 2 Problem 31.2-4

Based on Euclid's theorem:
- If b|a then gcd(a, b) = b.
- If a = bt + r, then gcd(a, b) = gcd(b, r)

EUCLID(a, b)
1. high = max(a, b) and low = min(a, b)
2. while (low > 0)
3. {
4.     t = $\lfloor high/low \rfloor$ ;
5.     r = high - low $\times$ t;
6.     high = low;
7.     low = r;
8. }
9. return high;

# 3 Problem 31.5-1

The given equations are : x $\equiv$ 4 (mod 5) and x $\equiv$ 5 (mod 11)
From the equations we have:
$x_1 = 4$
$n_1 = m_2 = 5$
$x_2 = 5$
$n_2 = m_1 = 11$
$n = n_1 \times n_2 = 5 \times 11 = 55$

Since $11^{-1} = 6(\mod 5)$ we have $m_1^{-1} = 6$
Similarly $5^{-1} = 20(\mod 11)$ thus $m_2^{-1} = 20$

$c_1 = m_1 \times (m_1^{-1} \bmod n_1) = 11 \times (6 \bmod 5) = 66$
$c_2 = m_2 \times (m_2^{-1} \bmod n_2) = 5 \times (20 \bmod 11) = 100$

$$\begin{aligned} \text{x} &\equiv \textstyle\sum_{i=1}^{2} x_i c_i \\ &\equiv (4 \times 66) + (5 \times 100) \pmod{55} \\ &\equiv 764 \pmod{55} \\ &\equiv 49 \pmod{55} \end{aligned}$$

Hence, solutions of the given equations are of the form $49 + 55 \times (n)$ for $n \geq 0$.

# 4 Problem 31.7-3

Let $P_A$ be the function corresponding to Alice's public key (e, n) and $M_1$ and $M_2$ be two messages. Thus, encrypted message $M_1$ is $P_A(M_1)$ and encrypted message $M_2$ is $P_A(M_2)$. Also, consider that

$$C_1 \equiv M_1^e \bmod \text{n and } C_2 \equiv M_2^e \bmod \text{n}$$

$P_A(M_1) = M_1 \bmod \text{n} = M_1^e \bmod \text{n}.$
$P_A(M_2) = M_2 \bmod \text{n} = M_2^e \bmod \text{n}.$
Multiplying,
$P_A(M_1)P_A(M_2) = (M_1^e \bmod \text{n})(M_2^e \bmod \text{n}) = (M_1M_2)^e \bmod \text{n} = C_1C_2.$

The multiplicative property can be exploited as follows. Assume that the attacker wants to find $M$ which is the decryption of ciphertext $C$. Note that the attacker has the knowledge of $C$ and of public key $P_A = (e, n)$.
Then, the attacker can select integer $r$ at random such that $r \in \mathbb{Z}_n$ and create a new "ciphertext" $\hat{C} = Cr^e$ mod n. If the attacker has a procedure for decrypting 1 percent of ciphertexts, he can obtain $\hat{M} = \hat{C}^d$.
$\hat{M} = \hat{C}^d = (Cr^e)^d = C^d r^{ed} = C^d r = Mr \bmod \text{n}.$
Thus, the attacker can compute $M = \hat{M} r^{-1} \bmod \text{n}.$

$Z_n$ is a finite multiplicative group of size n. It is given that the attacker knows 1 percent of the messages. That means, he knows $\frac{n}{100}$ messages. Using this, he has to calculate the remaining messages.

Let m be the number of messages decrypted so far and n be total numbers in $Z_n$. The routine below will decrypt the remaining messages.

**Algorithm**

```
for ( j=1 to n-m )
{
    for( i=1 to m )
    {
        R ← M_i
        Ĉ = C_j.r^e mod n

        / * decrypt Ĉ */
        M̂ = Ĉ^d mod n

        M_j = M̂.r^-1 mod n
        if (success)
        {
            temp = m;
            n = n - m;
            m = m + temp;
        }
    }
}
```