RSA con bloques:

No se generan aleatorios con el rango de bits, solo se cambiaron las funciones de cifrado y descifrado para soportar bloques y se agregó una función de tam para encontrar el tamaño de un ZZ

RSA.h

```
#ifndef RSA_H
#define RSA_H
#include <fstream>
#include <time.h>
#include <stdlib.h>
#include <string>
#include <sstream>
#include <mat.h>
#include <NTL/ZZ.h>
#include <iostream>

using namespace std;
using namespace NTL;


class RSA
{
    public:

        RSA();
        ZZ cifrar(string,ZZ,ZZ);
        string descifrar(ZZ);

        ZZ aleatorio(ZZ,string,ZZ);
        void generarclave();
        string cifrar_str(ZZ,ZZ);
        int descifrar_str(string,int);
        ZZ tam(ZZ);

        string alfabeto;
        ZZ publica;
        ZZ n;

        mat fun;

    private:
        ZZ privada;

};

#endif // RSA_H
```

RSA.cpp

```
#include "RSA.h"

RSA::RSA()
{

    alfabeto = "ABCDEFGHIJKLMNOPQRSTUVWXYZ,.-(
)abcdefghijklmnopqrstuvwxyz<>*1234567890";
    generarclave();

}
void RSA::generarclave()
{
    srand(time(NULL));

    //asignamos a un string los primos
    ifstream leer;
    leer.open("Primeros_1000000_Primos.txt");//hay 2 archivos más,
con los primeros 10000, 100000, 1000000 de primos.(cambiar el
numero por esos de ser necesario)
    string primos;
    getline(leer,primos);
    leer.close();

    //calculando la cantidad de primos en el string
    ZZ siz=conv<ZZ>("0");
    ZZ tam = conv<ZZ>(primos.size());
    for(int i=0;i<tam;i++)
        if(primos[i]=='.')siz++;

    //escojemos de manera aleatoria 2 primos
    ZZ num_rand = fun.mod(conv<ZZ>(rand()),siz);
    ZZ privada1 = aleatorio(num_rand,primos,tam);
    num_rand = fun.mod(conv<ZZ>(rand()),siz);
    ZZ privada2 = aleatorio(num_rand,primos,siz);

    //calculamos las variables n y fi de n
    n=privada1*privada2;
    ZZ fi_n=(privada1-1)*(privada2-1);

    //clave publica
    num_rand = fun.mod(conv<ZZ>(rand()),siz);
    publica=fun.mod(aleatorio(num_rand,primos,siz),n);
    while(fun.mcd(publica,fi_n)!=1)
        publica=fun.mod(aleatorio(num_rand,primos,siz),n);

    //clave privada
    privada=fun.inv_mult(publica,fi_n);
}
ZZ RSA::cifrar(string mensaje,ZZ clave,ZZ n){
```

```
    ZZ tam_n = tam(n);
    ZZ tam_alf = tam(conv<ZZ>(alfabeto.size()));
    string total,total2;

    //Posición en alfabeto y se guardan los bloques de posiciones
en total
    for(int i=0;i<mensaje.size();i++){
        ZZ pos = conv<ZZ>(alfabeto.find(mensaje[i]));
        total+=cifrar_str(pos,tam_alf);
    }
    ZZ tam_total = conv<ZZ>(total.size());
    //Agregar basura al final del string de ser necesario
    for(ZZ i=fun.mod(tam_total,tam_n-1);i<tam_n-1;i++)
        total.insert(total.size(),"0");

    while(total.size()>conv<ZZ>("0")){

    //Extraemos el bloque de tamaño n
    string str_aux = total.substr(0,conv<int>(tam_n)-1);
    total.erase(0,conv<int>(tam_n)-1);
    //Lo pasamos a ZZ para hacer la potencia
    istringstream istr_aux(str_aux);
    ZZ in;
    istr_aux >> in;
    in=fun.pow(in,publica,n);
    //Agregamos los 0 a cada bloque de ser necesario y se guarda
    str_aux=cifrar_str(in,tam_n);
    total2+=str_aux;
    }
    return total2;

}
string RSA::descifrar(ZZ mensaje){
    ZZ tam_n=tam(n);
    ZZ tam_total=conv<ZZ>(mensaje.size());
    ZZ tam_alf = tam(conv<ZZ>(alfabeto.size()));
    string contenedor1,contenedor2;
    while(tam_total>0){

        //Separamos por subconjuntos de tamaño n
        string str_aux = mensaje.substr(0,conv<int>(tam_n));
        mensaje.erase(0,conv<int>(tam_n));
        //Lo pasamos a ZZ para hacer la potencia
        istringstream istr_aux(str_aux);
        ZZ zz_aux;
        istr_aux >> zz_aux;
        zz_aux=fun.pow(zz_aux,privada,n);
        //Lo pasamos a string para guardarlo
        str_aux = cifrar_str(zz_aux,tam_n-1);

        contenedor1+=str_aux;
        tam_total=tam_total-tam_n;
    }

    //Extraemos los bloques de tamaño tam_alf
    tam_total = conv<ZZ>(contenedor1.size());
    for(int i=0;i<tam_total;i+=conv<int>(tam_alf)){
        int pos = descifrar_str(contenedor1,i);
```

```
        contenedor2 += alfabeto[pos];
    }

    return contenedor2;
}
ZZ RSA::tam(ZZ a){
    ZZ cont;//Iterador de las cifras de a
    if(a==conv<ZZ>("0"))cont = conv<ZZ>("1");//si es 0 tiene una
cifra
    else cont = conv<ZZ>("0");
    while(a!=conv<ZZ>("0")){a/=10;cont++;}
    return cont;
}
string RSA::cifrar_str(ZZ pos,ZZ tam_p){
    tam_p=tam_p-tam(pos);
    string cadena;
    while(tam_p!=conv<ZZ>("0")){
        cadena.insert(0,"0");
        tam_p--;
    }
    ostringstream aux;
    aux << pos;
    cadena.append(aux.str());
    return cadena;
}
int RSA::descifrar_str(string mensaje,int tam_n){

    int tam_alf = conv<int>(tam(conv<ZZ>(alfabeto.size())));
    string str_aux = mensaje.substr(tam_n,tam_alf);

    istringstream igstr_aux(str_aux);
    int pos;
    igstr_aux >> pos;

    return pos;
}
ZZ RSA::aleatorio(ZZ rand,string primos,ZZ tam){

        //escoje el primo en posición rand y lo guarda en base
        string base="";
        for(int i=0;rand!=0;i++){
                if(primos[i]=='.')
                        rand--;
                if(rand==0)
                    for(int j=i-1;primos[j]!='.';j--)
                        base=primos[j]+base;
            }

        //Converción de string a ZZ
        istringstream aux(base);
        ZZ primo;
        aux >> primo;
        return primo;

}
```